AFRL-RI-RS-TR-2011-251

NEW FRAMEWORKS FOR DETECTING AND MINIMIZING INFORMATION LEAKAGE IN ANONYMIZED NETWORK DATA

Johns Hopkins University

October 2011

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2011-251 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/ /s/

TANYA MACRINA
Work Unit Manager

WARREN H. DEBANY JR., Technical Advisor
Information Exploitation and Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
October 2011	Final Technical Report	March 2008 – April 2011

4. TITLE AND SUBTITLE

NEW FRAMEWORKS FOR DETECTING AND MINIMIZING INFORMATION LEAKAGE IN ANONYMIZED NETWORK DATA

5a. CONTRACT NUMBER
FA8750-08-2-0147

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)

Fabian Monrose

5d. PROJECT NUMBER
DHS5

5e. TASK NUMBER
TM

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Johns Hopkins University
W400 Wyman Park Bldg.
3400 N. Charles Street
Baltimore, MD 21218-2680

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/Information Directorate
Rome Research Site/RIGA
525 Brooks Road
Rome NY 13441

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2011-251

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The availability of realistic network data plays a significant role in fostering collaboration and ensuring U.S. technical leadership in network security research. Unfortunately, a host of technical, legal, policy, and privacy issues limit the ability of operators to produce data sets for information security testing. In an effort to help overcome these limitations, the Department of Homeland Security (DHS) has endeavored to create a national repository of network traces under the Protected Repository for the Defense of Infrastructure against Cyber Threats (PREDICT) program. A key technique used in this program to assure low-risk, high-value data is that of trace anonymization- a process of sanitizing data before release so that information of concern cannot be extracted. Indeed, many believe that proven anonymization techniques are the missing link that will enable cyber security researchers to tap real Internet traffic and develop effective solutions tailored to current risks.

15. SUBJECT TERMS

Anonymization, network data, deanonymize, Detect, Minimize, Information Leakage, Frameworks

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			TANYA MACRINA
U	U	U	UU	92	19b. TELEPHONE NUMBER (*Include area code*) N/A

Contents

List of Figures

List of Tables

Chapter 1

Summary

The availability of realistic network data plays a significant role in fostering collaboration and ensuring U.S. technical leadership in network security research. The challenge, however, has been that a host of technical, legal, policy, and privacy issues limit the ability of operators to produce datasets for information security testing. In an effort to help overcome these limitations, the Department of Homeland Security (DHS) has endeavored to create a national repository of network traces under the Protected Repository for the Defense of Infrastructure against Cyber Threats (PREDICT) program. A key technique used in this program to assure low-risk, high-value data is that of trace *anonymization*—a process of sanitizing data before release so that information of concern cannot be extracted. Indeed, many believe that proven anonymization techniques are the missing link that will enable cyber security researchers to tap real Internet traffic and develop effective solutions tailored to current risks.

Given the significant reliance on anonymized network traces for security research, in this project, we explored a principled approach to the problem of trace anonymization. **Specifically, we provided a framework for evaluating the risk associated with anonymization techniques and datasets**. Based on novel information-theoretic concepts, we showed how our techniques can be used by a security practitioner to evaluate the risk inherent in their choice of anonymization policy and technique—thereby minimizing the risks of deanonymization.

1.1 Goals

Recall that the main goals of the work supported by this contract were to enable techniques for detecting sources of information leakage in anonymized network traces. The developed tools are based off a new information-theoretic framework designed to provide data publishers with more useful tools for evaluating anonymity of objects within their network traces. More specifically, our goals were to provide (*1*) techniques for evaluating the level of anonymity achieved by an anonymization methodology, and to better evaluate the effects of arbitrary deanonymizations on other objects in the data (*2*) techniques for identifying possible sources of leakage and appropriate recommendations when safe anonymization is not possible (*3*) a framework for evaluating these and other policy decisions for network data publication in the context of current laws and popular

practices (*4*) an evaluation and risk mitigation plan for subsets of data in **PREDICT** (where allowed by policy and computationally feasible). An overview of our original process is given in Figure 1.1.

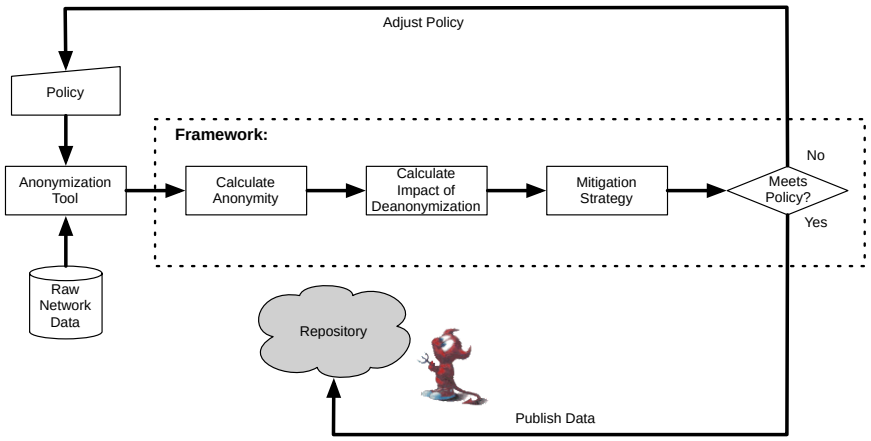


Figure 1.1: Overview

1.2 Deliverables

Our specific deliverables and contribution under this contract are summarized in the seven (7) peer-reviewed scientific publications presented in **Chapters 2—6.1**:

- As part of this project, **we provided techniques for evaluating the efficacy of network data anonymization techniques with respect to the privacy they afford [24]**. Specifically, we provided tools for simulating the behavior of an adversary whose goal is to deanonymize objects, such as hosts or web pages, within the network data. By doing so, we were able to quantify the anonymity of the data using information theoretic metrics, objectively compare the efficacy of anonymization techniques, and examine the impact of selective deanonymization on the anonymity of the data. We provided several concrete applications of our approach on real network data in the hope of underscoring its usefulness to data publishers. We elaborate further in § 2.
- **We compared the fields of microdata and network data anonymization to reveal the ways in which existing microdata literature may be applied to the network data anonymization problem [21]**. We further described challenges facing the development of robust network data anonymization methodologies that are grounded in the insights and lessons learned from microdata anonymization. Specifically, we examined the difficulties of clearly defining the privacy properties of network data due to its complex nature. In addition, we pointed out the necessity of utility measures in quantifying the extent to which anonymization may alter results obtained from analysis of the data. As a whole, our comparison between the fields of microdata and network data anonymization served to focus the attention of the research community on a holistic approach to network data anonymization that enables the type of collaboration necessary to further progress in the areas of network security research. We elaborate further in § 3.1.

- **We provided efficient and scalable tools for performing host-behavioral classification [20]**. These tools played a key role in our evaluation framework. More specifically, to explore the role of semantic and temporal characteristics of network data, we developed a novel behavioral distance metric for network hosts and compare its performance to a metric that ignores such information. Specifically, we proposed semantically meaningful metrics for common data types found within network data, showed how these metrics can be combined to treat network data as a unified metric space, and described a temporal sequencing algorithm that captures long-term causal relationships. In doing so, we brought to light several challenges inherent in defining behavioral metrics for network data, and put forth a new way of approaching network data analysis problems. We elaborate further in § 4.1.

- **We conducted several rigorous analyses [19, 20, 23, 24, 87, 89] of our techniques using very large dataset collected by our partners at University of Michigan and Merit, and made available as part of the** PREDICT **project**.

- **We provided a methodology for identifying sensitive data in packet payloads, motivated by the need to sanitize packets before releasing them (e.g., for network security/dependability analysis)**. Our methodology accommodates packets recorded from an incompletely documented protocol, in which case it will be necessary to consult a human expert to determine what packet data is sensitive. Since expert availability for such tasks is limited, however, our methodology adopts a hierarchical approach in which most packet inspection is done by less-trained "workers" whose designations of sensitive data in selected packets best match the expert's.

- **We provided the design and implementation of a tool [36] that implements an interactive method for packet trace anonymization**. We present an evaluation based on an *IRB-approved user study involving 15 professional network administrators* as workers. At a high level, our technique accomplishes our goals through a multistage process. Packets in the trace are first divided into contiguous tokens, each with a type. Stratified sampling is applied to these typed token sequences in order to select a fraction of the packets for further analysis, while minimizing the likelihood of excluding any particular packet type. These selected packets are clustered into groups with similar structure; intuitively (and ideally), these clusters correspond to packet formats. We then select representatives from each cluster, which we present to a worker in an aligned fashion so as to best reveal their common structures, a technique known to accelerate visual recognition of homogeneous structures. Our results showed that both clustering and alignment have a statistically significant, positive effect on their abilities to identify sensitive data, and that the effects of the two components are additive. We elaborate further in § 5.1.

- **We provided a legal toolchest summarizing the public policy implications of network trace anonymization, and a resulting framework for helping researchers and practitioner understand pertaining legal tort and statues guiding the collection and use of networking data**. This work is particularly important given the recent questioning of the legality of collecting network data [12, 60] and the ethical uses of such data [1]. The resultant **135-page document**, entitled "*Legal and Policy Toolchest for CyberSecurity*

Research and Development" is provided as a separated document. It includes a decisional worksheet, sample Memorandum of Agreement and Clauses, as well as sample Policies.

- Under our no-cost extension, we examined ideas for following the chain of custody of data in a virtualized environment. Specifically, we studied possible approaches for *tracking* accesses to objects that originate from disk, and capturing subsequent accesses to those objects in memory. *The goal was to investigate the application of a new data provenance approach to the sharing and use of networking and security data.* Specifically, we were interested in seeing how our approach might be used to help support the sharing of un-anonymized data with researchers using the **PREDICT** repository.

Chapter 2

Techniques for Evaluating Anonymized Network Data

2.1 Summary

The availability of realistic network data plays a key role in fostering new solutions to the latest network and security research problems. Unfortunately, due to the sensitive nature of the information that may be contained within such data, organizations are often reluctant to make network data available. Therefore, significant emphasis has been placed on developing techniques for *network data anonymization*, *i.e.*, the process of sanitizing data before release so that sensitive information cannot be extracted [63, 80, 82, 83]. Emboldened by the availability of these tools, the community has developed several data repositories to encourage sharing among researchers and practitioners [66]. However, when it comes to the efficacy of these anonymization tools, the devil is in the details [63].

Indeed, the anonymization systems relied upon today provide few, if any, guarantees on privacy. Moreover, they do not allow the publisher to quantify the risks involved with publishing a particular dataset. For the most part, progress in the development of anonymization systems has occurred primarily in reaction to the evolving threats to privacy or the need for greater utility (*e.g.*, [34, 61]). Given the reactionary nature of network trace sanitization, several works (including our own) showed that, in certain cases, it is possible to infer network topology [23], deanonymize public hosts [8, 9, 23, 41, 70], and identify web browsing behaviors [19, 41] despite the use of state-of-the-art anonymization. The existence of such attacks and the fact that there are, at present, no rigorous methods for evaluating the anonymization of network data may lead to a loss of confidence in anonymization techniques and a decrease in available network data.

Intuitively, it would seem that the problems faced in anonymizing network data can be overcome by applying advances in other domains. For instance, the problem of evaluating anonymity in network data would appear to be closely related to that of inference control in databases; each packet or flow in the network data might be considered as a row in a database, lending itself to the direct application of a vast body of work on privacy preserving techniques for databases. Unfortunately, the peculiar nature of network data makes the direct application of these techniques problematic. Most notably, when dealing with network data, it is difficult to know *a priori* which fields should be considered sensitive. Finding such channels of information leakage lies at the heart

of the network data sanitization problem. Furthermore, some of the most sensitive information in the network data, such as web browsing activities, are encoded in the distributions of values over several packets or flows. These differences warrant the development of new techniques tailored to evaluating network data.

In this part our the contract, we focused on addressing the aforementioned problems by providing an analysis of anonymized network data as the simulation of an adversary whose goal is to distinguish the true identity of an anonymized object, such as a host, from among all possible unanonymized identities. Generally speaking, attacks on network data anonymization proceed as follows. The adversary approximates the expected feature distributions (*e.g.*, distribution of local port numbers) for unanonymized objects by using public information sources, such as DNS records or web search engines. She then calculates the feature distributions for each of the anonymized objects, and compares them to the approximated distributions for the unanonymized objects. If a one-to-one mapping exists between the anonymized and unanonymized distributions, then she can infer the identity of the object. This general algorithm for the adversary's behavior describes a broad and meaningful class of inference attacks, including several attacks against anonymized network data [8, 9, 19, 23, 41, 70].

Our approach simulates such an adversary by comparing objects from the unanonymized and anonymized data directly, thereby assuming that she has perfect knowledge of the unanonymized distributions. To analyze the anonymity of an object, we calculate the feature distributions for that anonymized object, and compare them to the feature distributions of all objects in the unanonymized data that are of the same type (e.g., host objects). From this comparison, we derive a probability distribution over the object's possible unanonymized identities. To quantify the anonymity of the object, we calculate the information entropy of this derived probability distribution. Furthermore, we model the auxiliary information gained by the adversary from meta-data provided by the data publisher, and from deanonymization of objects within the data.

Our contributions were in showing that through the use of entropy metrics, publishers can identify objects that are at risk of deanonymization, and can objectively compare the performance of various anonymization systems and policies. Moreover, by simulating the adversary's auxiliary information, a publisher can examine the impact of deanonymization on the anonymity of other objects. To underscore the utility of our approach, we provided several concrete applications of this analysis to real network data.

2.2 Introduction

The availability of realistic network data plays a key role in fostering new solutions to the latest network and security research problems. Unfortunately, due to the sensitive nature of the information that may be contained within such data, organizations are often reluctant to make network data available. Therefore, significant emphasis has been placed on developing techniques for *network data anonymization*, *i.e.*, the process of sanitizing data before release so that sensitive information cannot be extracted [63, 80, 82, 83]. Emboldened by the availability of these tools, the community has developed several data repositories to encourage more sharing among researchers and practitioners [26, 66, 78]. However, when it comes to the efficacy of these anonymization tools, the devil is in the details [63].

Indeed, the anonymization systems relied upon today provide few, if any, guarantees on privacy.

Moreover, they do not allow the publisher to quantify the risks involved with publishing a particular dataset. For the most part, progress in the development of anonymization systems has occurred primarily in reaction to the evolving threats to privacy or the need for greater utility (*e.g.*, [34, 61]). Given the reactionary nature of network trace sanitization, it comes as no surprise that several recent works have shown that, in certain cases, it is possible to infer network topology [23], deanonymize public hosts [8, 9, 23, 41, 70], and identify web browsing behaviors [19, 41] despite the use of state-of-the-art anonymization. The existence of such attacks and the fact that there are, at present, no rigorous methods for evaluating the anonymization of network data may lead to a loss of confidence in anonymization techniques and a decrease in available network data.

Intuitively, it would seem that the problems faced in anonymizing network data can be overcome by applying advances in other domains. For instance, the problem of evaluating anonymity in network data would appear to be closely related to that of inference control in databases (*e.g.*, [15, 32, 33, 75]); each packet or flow in the network data might be considered as a row in a database, lending itself to the direct application of a vast body of work on privacy preserving techniques for databases. Unfortunately, the peculiar nature of network data makes the direct application of these techniques problematic. Most notably, when dealing with network data, it is difficult to know *a priori* which fields should be considered sensitive. Finding such channels of information leakage lies at the heart of the network data sanitization problem. Furthermore, some of the most sensitive information in the network data, such as web browsing activities, are encoded in the distributions of values over several packets or flows. These differences warrant the development of new techniques tailored to evaluating network data.

In this paper, we frame the analysis of anonymized network data as the simulation of an adversary whose goal is to distinguish the true identity of an anonymized object, such as a host, from among all possible unanonymized identities. Generally speaking, attacks on network data anonymization proceed as follows. The adversary approximates the expected feature distributions (*e.g.*, distribution of local port numbers) for unanonymized objects by using public information sources, such as DNS records or web search engines. She then calculates the feature distributions for each of the anonymized objects, and compares them to the approximated distributions for the unanonymized objects. If a one-to-one mapping exists between the anonymized and unanonymized distributions, then she can infer the identity of the object. This general algorithm for the adversary's behavior describes a broad and meaningful class of inference attacks, including several attacks against anonymized network data [8, 9, 19, 23, 41, 70].

Our approach simulates such an adversary by comparing objects from the unanonymized and anonymized data directly, thereby assuming that she has perfect knowledge of the unanonymized distributions. To analyze the anonymity of an object, we calculate the feature distributions for that anonymized object, and compare them to the feature distributions of all objects in the unanonymized data that are of the same type (*e.g.*, host objects). From this comparison, we derive a probability distribution over the object's possible unanonymized identities. To quantify the anonymity of the object, we calculate the information entropy of this derived probability distribution. Furthermore, we model the auxiliary information gained by the adversary from meta-data provided by the data publisher, and from deanonymization of objects within the data.

The main contributions of this paper are in showing that through the use of entropy metrics, publishers can identify objects that are at risk of deanonymization, and can objectively compare the performance of various anonymization systems and policies. Moreover, by simulating the ad-

versary's auxiliary information, a publisher can examine the impact of deanonymization on the anonymity of other objects. To underscore the utility of our approach, we provide several concrete applications of this analysis to real network data.

2.3 Methods, Assumptions, and Procedures

The analysis techniques presented in this paper make heavy use of concepts from probability and information theory. For ease of exposition, we first review the concepts and definitions used throughout the remainder of this paper.

2.3.1 Probability Distributions and Random Variables

Our analysis is founded on the examination of joint and marginal probability distributions computed from the set of values found in the various fields of the network data. In general, we can consider a *random variable* X that describes the probability distribution over values from a single field in the network data. A *joint distribution* on fields represented by random variables X and Y describes the probability distribution of the values from X and Y that occur together, and the probability mass function for the joint distribution is denoted as $p(x, y)$, where x and y are values taken from the sample spaces of X and Y, respectively. A joint distribution can be created from an arbitrary number of random variables by generalizing the definition accordingly. Similarly, a *marginal distribution* on a field represented by the random variable X would describe the probability distribution of the values from X alone, and the probability mass function for the marginal distribution is denoted as $p(x)$. Since the fields found in network data contain discrete values, we represent marginal distributions as a histogram where each bin represents the probability mass for a single value in the sample space of the distribution. Likewise, a discrete joint distribution on random variables X and Y would be described such that each bin represents the probability mass for the pair of values x and y taken from their respective sample spaces, and so on for increasing numbers of random variables.

In some cases, the values in the network data may change slightly, although their underlying meanings are equivalent. For instance, if two TCP connections send the exact same data, then their histograms of packet sizes should be exactly the same (*i.e.*, their bin values and amplitudes should be the same). However, various changes could occur, such as TCP retransmissions, which skew the histograms. One way of overcoming this problem is to implement a smoothing technique on the distribution [16]. In practice, any number of smoothing strategies may be applicable, but for our analysis we implement a very simple smoothing strategy that proceeds as follows. For a random variable Z representing the distribution to be smoothed, we find the standard deviation of Z and initially create one bin for each value such that the bin represents the range of that value plus/minus the standard deviation. Those bins with overlapping ranges are merged into a single bin such that the smoothed histogram generated contains bins representing the probability mass for each permissible range of values.

2.3.2 Information Entropy and Mutual Information

The concept of *information entropy* [25] is central to our analysis in that it provides a single value that quantifies the amount of uncertainty in a random variable, and acts as an intuitive indicator

of anonymity [29, 77]. The entropy of a random variable X represents the amount of information gained by learning the outcome of X, and is calculated as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{2.1}$$

The entropy of X takes its minimum value, zero, when all probability mass is centered on a single value, and it takes its maximum value, $\log N$ where N is the size of the sample space, when all values are equiprobable.

Likewise, the *mutual information* [25] between two random variables, X and Y, represents the amount of information gained about one variable by learning the outcome of the other, and is calculated as:

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{2.2}$$

The mutual information between two random variables is symmetric, and takes its minimum value of zero when the two variables are statistically independent. Naturally, since mutual information indicates the information shared by the two variables, it is limited by the minimum of their information entropies. To transform mutual information into a more intuitive notion of correlation, we can normalize it by its maximum value, as shown in Equation 2.3. The *normalized mutual information* takes values near one when the two variables are heavily correlated, and values near zero when they are independent. This normalized version makes it far more intuitive to set a defining threshold between correlation and independence.

$$\overline{I(X;Y)} = \frac{I(X;Y)}{min(H(X), H(Y))} \tag{2.3}$$

2.3.3 L1 Similarity Metric

In order to appropriately quantify the anonymity of the network data, we require a similarity metric to compare distributions to one another. Since we focus our analysis on discrete distributions represented by histograms, an intuitive choice for this similarity metric is the *L1 similarity*. The L1 similarity represents the difference between the maximum L1 distance minus the sum of the absolute differences between each of the corresponding bins in the two distributions, X and Y, as shown in Equation 2.4. In general, the bins in X and Y correspond to one another if they represent the same value.

$$sim(X,Y) = 2 - \sum_{z \in \mathcal{X} \cup \mathcal{Y}} |P(X = z) - P(Y = z)| \tag{2.4}$$

The L1 similarity takes its maximum value, two, when both distributions are identical, and its minimum value, zero, when the sample spaces of the distributions are completely disjoint.

In our analysis, the L1 similarity calculation is complicated by the fact that the anonymization process may have permuted or otherwise changed the values in the anonymized data (*e.g.*, shuffling port numbers). In this case, the values that the bins represent will no longer map directly between the unanonymized and anonymized distributions. We will reexamine this issue in Sections 2.3.4 and 2.6.1, and show how to use a mapping relation to correctly simulate the adversary's information about the correct mappings in this L1 similarity calculation.

2.3.4 Adversarial Model

The underlying principle of our analysis is a realistic simulation of an adversary whose goal is to deanonymize the network data. For clarity, we explicitly define the goal of our simulated adversary, state our assumptions on the adversary's power, and describe how those assumptions relate to real-life adversaries.

The Adversary's Goal

In our analysis, an *object* is characterized by a set of distributions on the features of the network data (*i.e.*, fields or groups of fields), which represent its presence in the data. Thus, an *(un)anonymized object* is an object with its distributions calculated from the (un)anonymized version of the network data. Given an anonymized object and a set of unanonymized objects, the adversary's goal is to create a mapping between the anonymized object and its unanonymized counterpart by comparing the feature distributions of the anonymized object to all unanonymized objects.

This definition of the adversary's goal maps closely to that of a real-life adversary mounting an inference attack. In both cases, the adversary compares feature distributions to infer the identity of an anonymized object. The primary difference between the two is that the real adversary must approximate the set of unanonymized objects and their distributions, while our simulated adversary has exact information about the objects and their distributions, drawn directly from the unanonymized network data. In doing so, we perform a *worst-case* analysis of the indistinguishability of the objects.

Auxiliary Information

Beyond the feature distributions themselves, the adversary also has access to *auxiliary information*, which she can use to refine the accuracy of her inferences. One of the primary problems for the adversary in deanonymizing the data lies in accurately comparing the anonymized distributions to the unanonymized distributions. The anonymization process is intended to make it difficult for the adversary to directly compare the distributions, and so a relation must be applied which maps the anonymized values to their potential unanonymized counterparts. The auxiliary information is used to refine this mapping relation.

Object Type	Definition
Host	Distinct Local IP
Web Page	Distinct Local IP and (Interarrival time $<$ n seconds and Remote Port = 80)
User	Distinct Local IP and (User,Distinct IP,Time in Authentication Log)

Table 2.1: Example definitions used to group records into objects

One piece of auxiliary information available to a real-life adversary is expert knowledge of the network data semantics. From knowledge of the semantics, the adversary can examine the relationships among the fields in the network data to select features that best describe the objects. In our simulation, the fields in the network data are annotated with their semantic type, and

an automated feature selection algorithm is used to extract the features upon which we perform our analysis. The annotation and feature selection process is discussed in Sections 2.4 and 2.5, respectively.

The real-life adversary also has access to meta-data, which the data publisher provides to researchers to explain the context of the anonymized network data [65]. This meta-data might describe which packets had incorrect checksums, the anonymized IP prefixes for the network where the data was collected, or the types of anonymization (if any) applied to each field in the data. Information, such as the anonymized prefixes, can be learned directly from the meta-data, which allows the adversary to refine the mapping relation and increase the accuracy of her inference attack. Also, other information from the meta-data, such as the type of anonymization used, helps in defining a set of algorithms that the adversary can use to extract additional refinements to the mappings after objects are deanonymized. The meta-data information is added to our simulation as annotations to the data, and we simulate the adversary's use of the learned information by implementing a mapping relation, which is described in greater detail in Section 2.6.1.

2.4 Annotating the Data

For our purposes, network data consists of n *records*, where each record is a tuple of m *fields* derived from a single packet or network flow [17]. A packet record might contain fields taken from the link, network, and transport layer headers of a packet. Flow records, on the other hand, contain fields which summarize information about all packets that were transmitted during a single session between two endpoints.

Network data can contain records generated due to the presence of several types of objects. At a high level, we might define a host object by the records sent or received by a single host, a web page object might consist of those records created due to the download of a specific web page, or a user object could be defined by all of the records sent or received by a host while a specific user was logged on to that system. To define an object in our analysis, the data publisher describes constraints on the fields within the network data (*e.g.*, each IP address from a given prefix is to be considered a host object). Clearly, the definition of an object is limited only by the data publisher's knowledge of the object when the data was collected. Some objects, like hosts, can be defined with relatively little information by considering each unique IP address from the data publisher's administrative domain to be a host object. Conversely, defining objects like web pages or users may require extra information, such as web proxy or authentication logs. Example object definitions can be found in Table 2.1.

Original		Inter-record		Intra-record		
Local IP	Remote IP	Local IP	Remote IP	Original Local IP, Original Remote IP	Original Local IP, Inter-record Local IP	...
192.168.0.10	192.168.0.250	0.0.0.0	0.0.0.0	0.0.0.240	192.168.0.10	
192.168.1.50	192.168.10.20	0.0.1.56	0.0.10.238	0.0.11.38	192.168.0.10	...
⋮	⋮	⋮	⋮	⋮	⋮	

Figure 2.1: Example data after creating inter- and intra- record fields with semantic type of IP and distance measure of XOR

Once the objects are defined, the data publisher describes the characteristics and semantics of each field in the data. This annotation of the data ensures that the analysis treats each field appropriately with respect to the values it contains. For our purposes, three annotations on the data are required, which are also required by current anonymization tools [63, 80]. First, the publisher provides the IP prefixes of the local networks that appear in the data, along with their anonymized counterparts. This annotation ensures that our techniques only analyze the anonymity of those objects within the publisher's administrative domain, and allows us to reformat the data in terms of local and remote, rather than source and destination. Second, the publisher lists the semantic types for each field, including a distance measure for that semantic type and the type of smoothing applied (if any). For instance, fields containing IP addresses would have the type `IP`, with a distance operator of XOR and no smoothing. By assigning types to the fields, we can automatically extract semantically meaningful information about how these fields relate to one another. Third, the data publisher lists the type of anonymization applied to each field. The anonymization type, such as prefix-preserving IP anonymization or deterministic mapping, dictates the algorithm to be used in reversing the anonymization and augmenting the adversary's auxiliary information. Note that in many cases, the semantic and anonymization types can be set to use default settings for the given field, thereby limiting the burden on the data publisher.

2.5 Feature Extraction and Selection

Many of the fields in the network data share complex and semantically meaningful relationships which can be considered as *features*. For instance, port scanning behavior that might fingerprint a host is visible in the way the destination port numbers change from record to record. Also, combinations of fields may produce a unique distribution where the constituent marginal distributions on those fields are not unique. Before applying our analysis techniques, it is important to examine the semantically meaningful relationships among fields, and extract the features that best represent those relationships.

2.5.1 Discovering Semantically Meaningful Relationships

The semantic types of the fields are used to characterize their relationships within the data, both within a record (*intra-record*) and across two or more records (*inter-record*). Initially, each of the n records in the network data contains m fields annotated with their semantic type. For each of the m initial fields, a new inter-record field is added to each row that inherits the type of the initial field. Likewise, we create a new intra-record field for each pair of fields, both initial and inter-record, with the same semantic type. The newly created intra-record field inherits the semantic type of the fields used to create it.

To populate the value for an intra-record field in record k, we use the distance measure defined for its semantic type and calculate the distance between its two constituent fields in record k. For the value of an inter-record field in record k, we compute the distance between the value of its initial field at row k with the same field's value at row $k - i$ (in our case, $i = 1$), where the records are sequential and all belong to a single object. For the inter-record field values at record $k = 0$, we set the distance to be zero since there is no predecessor record.

Figure 2.1, for instance, shows the resulting network data after inserting the inter- and intra-record fields. The data initially contains the Local and Remote IP fields. Then two inter-record

fields are created to calculate the distance between values in the Local and Remote IP fields across consecutive rows (*i.e.*, $i = 1$), respectively. Finally, $\binom{4}{2}$ intra-record fields are created for every pair of initial and inter-record fields with the same type to calculate the distance between values within the same record.

We note that the network data may contain some classes of subliminal channels that are not covered by our discovery process, such as complex active probing attacks [5, 79]. Discovering all possible subliminal channels in network data quickly becomes computationally infeasible since it requires the examination of *all* inter-record and intra-record relationships, whether they are semantically meaningful or not. Instead, we focus our analysis on a much narrower subset of these relationships that represent typical areas of information leakage that naturally occur within network data.

2.5.2 Feature Selection

When examined in isolation, the distribution of values in each of the fields may be similar among the objects being analyzed. However, when examined in aggregate these fields may be unique and could be used to distinguish the object. To see why, consider the case where two hosts have a variety of local port and remote IP address values in their respective records. Taken individually, the distributions on the local port and remote IP values for these hosts will be similar in an information theoretic sense. However, if one host always communicates with a single remote IP on a given port, while the other host communicates with a variety of remote IPs on that same port, then the two hosts will be distinguishable.

A na¨*i*ve solution to this problem is to analyze all possible joint distributions on these fields, but as with the inter-record relationships, this will quickly lead to an intractable number of distributions to examine for each object. Instead, we select those joint distributions whose constituent marginal distributions are heavily correlated with one another, since the scenario described above can only occur when the marginal distributions are correlated to some degree. Moreover, if two fields are highly correlated, then analyzing both fields for anonymity is redundant, and thus we can reduce the computational requirements of the analysis by examining the joint distribution instead.

Anonymization Type	Deanonymized Values		aux_{t+1} Relation
	Anonymized	Unanonymized	
Deterministic Mapping	port 150	port 80	$150 \rightarrow 80$
CryptoPAn [34]	200.120.10.10	128.2.250.220	$200.120.10.6 \rightarrow 128.2.250.208/29$
Pang *et al.*[63]	200.120.10.10	128.2.250.220	$200.120.10.6 \rightarrow 128.2.250.0/24$

Table 2.2: Examples of reversing anonymization and updating aux_t relation

The process of selecting a mutually independent set of features proceeds as follows. For each pair of fields (including the newly created inter- and intra-record fields), we calculate the normalized mutual information between their marginal distributions, as shown in Section 4.3. If the normalized mutual information is above some threshold (0.99 in our evaluation) we group the two fields together. We further combine groups of fields if they share at least one field in common. Finally, we create a singleton group for each field which has not been added to any other group. When the feature selection process completes, we arrive at several groups of fields where the fields within a group are transitively correlated to one another, and fields in different groups are mutually

independent according to our threshold of correlation. We take this set of mutually independent groups of fields to be the *features* upon which we perform our analysis.

2.6 Analyzing Network Data Anonymity

The primary goal of our analysis is to quantify the degree to which anonymized objects are distinguishable within the provided network data. To perform this analysis, we compare the features of an anonymized object to those of all unanonymized objects using the L1 similarity (see Section 4.3), and use this comparison to derive a probability distribution on the potential true identities of the object. Finally, we use the resultant probability distribution to calculate an entropy measure that quantifies the distinguishability of the anonymized object. However, it is important that the simulation of the adversary's behavior accurately reflect the auxiliary information available to the adversary at each step in the analysis.

2.6.1 Modeling Auxiliary Information

The auxiliary information available to the adversary can be modeled as a relation $aux_t : A \to \mathcal{P}(U)$, where t is the current time step in our simulation of the adversary, A is the set of anonymized values, and $\mathcal{P}(U)$ is the power set of unanonymized values. Thus, for some anonymized input, aux_t outputs a set of one or more potential unanonymized values given the information available to the adversary at time t. At the start of the analysis ($t = 0$), the adversary only has information gained from annotations on the data. Thus, for fields with no anonymization applied to them, $aux_{t=0}$ will output the same value that was used as input, since both the anonymized and unanonymized values are the same. For fields with deterministic mappings, the relation would return all unanonymized values for that field since the adversary does not know which is the correct mapping for the anonymized input. For example, in the case of prefix-preserving IP anonymization, the relation would output a set of unanonymized IPs that is consistent with the adversary's knowledge of the local enterprise IP prefix.

As the analysis progresses, we simulate the knowledge gained by the adversary after deanonymizing objects in the data. Upon deanonymizing an object, the adversary can reverse the anonymization by running an algorithm that compares anonymized and unanonymized values for the deanonymized object. A new relation aux_{t+1} is generated from the previous relation at time t and the newly learned information from the latest deanonymization. For instance, if a field were anonymized using deterministic mapping, the adversary learns the mapping between pairs of anonymized and unanonymized values, and the new relation is created such that those pairs have a one-to-one mapping. For prefix-preserving IP anonymization, the adversary checks the length of the prefix match between the anonymized IPs of the deanonymized object and all remaining anonymized IPs. The new relation then requires that those remaining anonymized IPs match their unanonymized counterparts up to the length of the calculated longest prefix match. Concrete examples of learning information from deanonymization are given in Table 2.2.

The auxiliary information relation is used to constrain our comparison between anonymized and unanonymized values in two ways. First, as mentioned in Section 4.3, L1 similarity requires the existence of a mapping between the bins of the distributions being compared. For comparison between anonymized and unanonymized values, the mapping between bins must respect the information available to the adversary in the relation aux_t. From among all of the possible mappings

allowed by aux_t, we choose the mapping that provides the highest L1 similarity. Second, objects are defined by various constraints on the fields in the network data (*e.g.*, local IPs defining host objects). In order to provide a correct analysis of the anonymity of an object, each anonymized object should only be compared to those unanonymized objects for which a mapping is possible given the relation aux_t.

2.6.2 Object Anonymity

Our analysis begins by examining the anonymity of each anonymized object in isolation, called *object anonymity*, where we consider only the information found in the data annotation. To perform this analysis, we begin with the relation $aux_{t=0}$ and the ℓ feature distributions $F_{i=1...\ell}$ identified using the technique from Section 2.5.2. For each feature distribution F_i, we calculate the L1 similarity between distribution $F_{i,A}$ for anonymized object A and distribution F_{i,U_j} for all unanonymized objects U_j. We use the similarity values as a count to approximate a probability distribution on the true unanonymized identity of A with respect to feature F_i as:

$$P(F_{i,A} = F_{i,U_j}) = \frac{sim(F_{i,A}, F_{i,U_j})}{\sum_{k \in aux_t(A)} sim(F_{i,A}, F_{i,U_k})} \tag{2.5}$$

Notice that this probability distribution considers only those objects present within the data, which follows from our assumption that the adversary knows the set of objects in the data and their exact unanonymized distributions. From this probability distribution on the identity of A, we can calculate the entropy of A with respect to feature F_i using Equation 5.4. In practice, there are many cases where a single unique distribution may lead to deanonymization. The entropy measures calculated for each of the feature distributions can be used to examine cases of specific attacks on those distributions, and whether the anonymization policy applied to those distributions was effective in providing anonymity.

We also calculate the overall entropy of an object from the entropy of each of its constituent feature distributions. Recall that the set of features produced from the feature selection process are *mutually independent* according to the threshold on the normalized mutual information. Therefore, given the entropy for each of the features $F_{i=1...\ell}$, we calculate the overall entropy of the identity of A as:

$$H(A) = \sum_{i=1}^{n} H(F_{i,A}) \tag{2.6}$$

Since we are summing the entropy value of each feature, the overall entropy value for a single object can be as high as $\ell \log N$, where ℓ is the number of features and N is the number of objects. The overall entropy of the object assumes that learning the identity of an object requires the adversary to learn the true identity for *all* feature distributions of that object. Though this may not be the case in many attack scenarios, we believe that the overall entropy values are still beneficial in understanding the general risk of deanonymization, and performing comparisons among different anonymization techniques.

2.6.3 Conditional Object Anonymity

Though the object anonymity provides useful information about the anonymity of the data, it does not capture the resilience of an anonymized dataset to deanonymization. For instance, it may be

possible that the anonymity of objects within the data appears reasonable, but upon deanonymizing a single object the adversary learns enough information to undermine the remainder of the data. Therefore, it is imperative that data publishers have a method of detecting and quantifying the effects of deanonymization so that comparisons can be made among various anonymization techniques. In order to achieve this, we need to consider the anonymity of hosts conditioned upon the deanonymization of other objects, which we call *conditional object anonymity*.

To perform our conditional anonymity analysis, we first begin with the object anonymity analysis described above using the relation $aux_{t=0}$. We then choose the object with the lowest entropy value, and assume that it has been deanonymized by the adversary. In doing so, we create a new relation aux_{t+1} from the information learned from the deanonymization, and again perform the same analysis as the object anonymity case using aux_{t+1} to constrain the mapping of anonymized values. We continue this process iteratively, choosing the lowest entropy object at each step in the process, learning its information, and augmenting the mapping relation. The process completes when all anonymized objects have been deanonymized. The sequence of deanonymizations provides the data publisher with a notion of the adversary's strategy for deanonymizing the data, and allows different anonymization techniques to be compared to better understand their performance after deanonymization occurs.

2.7 Results and Discussion

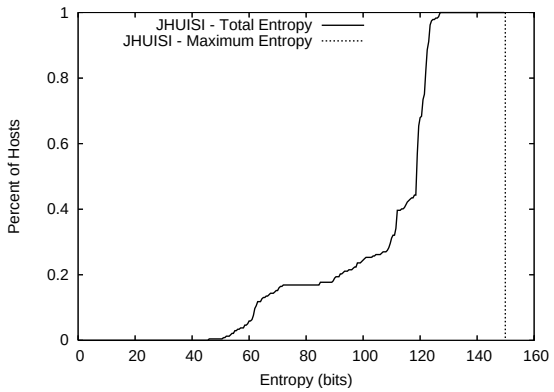


Figure 2.2: CDF of the total entropy of host objects, assuming all features are considered independently

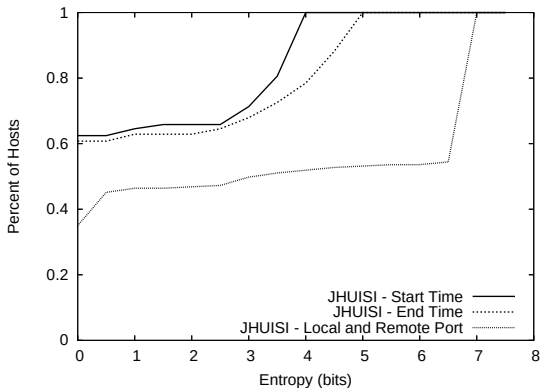


Figure 2.3: CDF of the three worst features

To underscore the utility of our analysis techniques, we examine several concrete uses of those techniques in quantifying the anonymity of network data. We apply our techniques to a network trace recorded over a period of 24 hours at the edge of the Johns Hopkins University Information Security Institute (JHUISI) network. That network contains three subnets with a total of 237 hosts present in the data. For our analysis, we treat each of the 27,753 flows in the data as a single record with 19 features derived (as described in Section 2.5) from the original fields: `start time`, `end time`, `local IP`, `local port`, `local size`, `remote IP`, `remote port`, `remote size`, and `protocol`. In our data, the `remote size` and `local size` fields are the sum of the packet

sizes sent to the remote and local hosts in the given flow, respectively. Our subsequent analysis focuses on the anonymity of host objects in the network data, where each distinct local IP address is treated as a different host object. In what follows, we show how a data publisher can (*i*) examine which objects may risk deanonymization due to unique distributions, (*ii*) objectively compare the anonymity provided by different anonymization techniques, and (*iii*) examine the impact of selective deanonymizations on the anonymity of other objects in the data.

2.7.1 Quantifying overall anonymity

One of the most obvious applications of our analysis is in quantifying the anonymity of the objects in a particular dataset. Figure 2.2 shows the cumulative distribution function (CDF) of the total entropy for the host objects in the dataset, as calculated by Equation 2.6. A CDF, such as the one provided here, allows the data publisher to quickly and intuitively quantify the overall anonymity of the objects in the dataset. The graph shows, for example, that a substantial fraction of hosts in the data have relatively low entropy compared to the maximal entropy—assuming, of course, that all features are independent of one another. An equally useful view is the CDF of the lowest entropy features across all hosts in the dataset, as shown in Figure 2.3. This view provides the data publisher with a method for verifying the soundness of the chosen anonymization policy for each feature. For instance, Figure 2.3 shows that the remote and local port feature provides distinguishing information for many of the hosts in the data, and as such, it may be prudent to re-examine the anonymization policy for these features.

2.7.2 Comparative analysis

Another important contribution of this type of analysis is that it allows for an objective comparison of anonymization techniques and the impact that selective deanonymizations have on their soundness. As an example, we compare the prefix-preserving IP anonymization system of Pang *et al.*[63] to the CryptoPAn system [34], as applied to the JHUISI dataset. For context, recall that the CryptoPAn system is strictly prefix-preserving, meaning that if two unanonymized IP addresses share a k-bit prefix, then so will their anonymized counterparts. The anonymization system of Pang *et al.*, however, uses a pseudo-random permutation to anonymize subnet and host portions of the IPs separately. Therefore, the approach of Pang *et al.*only guarantees that two IPs in the same unanonymized subnet will also be in the same anonymized subnet, but all other prefix relationships among the IPs are broken. In this comparison, we examine the average entropy across all hosts at each step in the conditional anonymity analysis, as shown in Figure 2.4. The graph illustrates several interesting properties of these two anonymization techniques, which allow a data publisher to make informed decisions on their use. Most notably, the graph shows that the approach of Pang *et al.*provides greater privacy than the CryptoPAn approach, especially after a few deanonymizations (three, in this case) have occurred.

Additionally, Figure 2.4 shows an interesting pattern of increasing average entropy followed by an abrupt drop, which occurs repeatedly for CryptoPAn, and less often for the approach of Pang *et al.*To see why this pattern emerges, it is instructive to recall how we model the process of deanonymization. In particular, recall that at each iteration of the conditional anonymity analysis we choose the anonymized host with the lowest entropy and deanonymize it, thus revealing its true identity. This deanonymization can result in two possible outcomes for the adversary's auxiliary

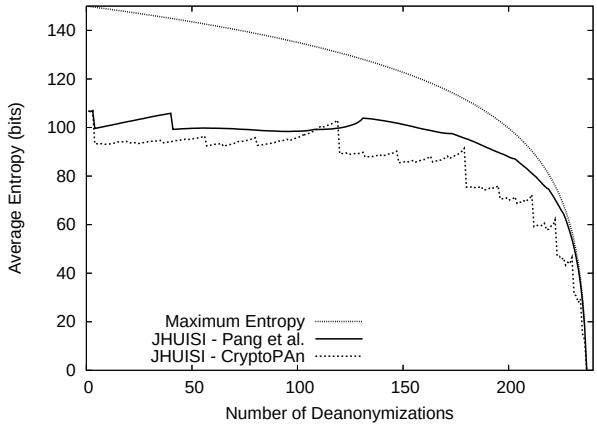


Figure 2.4: Comparison of Pang *et al.*anonymization to CryptoPAn

information. In the case where the deanonymization results in learning prefix information about a substantial number of hosts, the average entropy will drop significantly due to the change in entropy for each of the affected hosts (*e.g.*, iteration 40 for Pang *et al.*and 120 for CryptoPAn). When the deanonymization provides no information or information on very few hosts, the result is an increase in the average entropy (*e.g.*, iterations 4–39 for Pang *et al.*and 81–119 for CryptoPAn). This somewhat unintuitive increase occurs in part due to the removal of very unique hosts that skew the average entropy downward and the fact that we gain little or no additional information about the remaining hosts. Once all unique hosts are exhausted and the adversary learns all available information about the anonymized subnets, the average entropy decreases in a smooth logarithmic curve, as seen in the Pang *et al.*plot. By contrast, the average entropy under CryptoPAn continues the pattern of increased entropy followed by drastic drops because additional information about the prefix is continually learned throughout the deanonymization process.

2.7.3 On the impact of selective deanonymizations

Given the continued threat to data privacy, one intriguing benefit of our analysis is that it allows data publishers to explore the effect of selective deanonymizations. As a concrete example, we apply our analysis techniques to the JHUISI dataset in the context of a recently discovered behavioral profiling attack [23]. In that attack, the adversary uses public information to estimate the services provided by hosts (as indicated by port numbers) within the network where the anonymized data was captured. By comparing the estimated services with those indicated in the anonymized data, the adversary can reveal the host's identity.

To evaluate the impact of this attack on the data, we examine the entropy for the local and remote port feature for each host in the anonymized data with publicly available information. For our purposes, we simply use the number of references found in a popular search engine for each hostname in our network data to approximate its available public information. Table 2.3 shows each of the hosts with at least one reference, along with their entropy values for the local and remote port feature. Notice that there is a substantial gap between the first three hosts and all others in Table 2.3 in terms of their public references and entropy values. The results indicate that

Hostname	Local, Remote Port Entropy	Search Engine References
simnet	0.0	77
spar	0.0	71
skdnssec	0.004	32
mirror	3.50	10
cable	6.707	4

Table 2.3: Entropy and references in a popular search engine for hosts in the example dataset

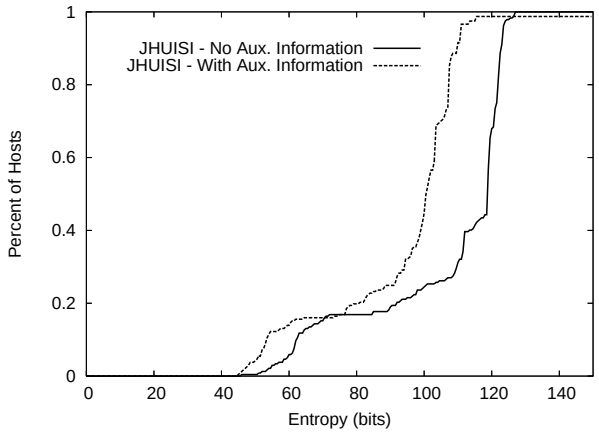


Figure 2.5: CDF of the the total entropy with and without deanonymized servers in the auxiliary information

simnet, *spar*, and *skdnssec* are at risk of deanonymization from the behavioral profiling attack, and, in fact, these were the same three hosts that were deanonymized in previous work [23].

Armed with insights regarding the hosts likely to succumb to the behavioral profiling attack, we can examine the effect that these deanonymizations would have on the rest of the data. We do so by assuming these hosts have been deanonymized and are now in the adversary's auxillary information, then repeat the computation in Section 2.6.3. Specifically, for our JHUISI data anonymized with the Pang *et al.*anonymization system, we take the anonymized values associated *simnet*, *spar*, and *skdnssec*, add their correct mappings to aux_t as if they have been deanonymized, and then recompute the total anonymity of the dataset.

Figure 2.5 shows the CDF for the JHUISI dataset without deanonymizations compared with the case where we assume the three hosts have been deanonymized and added to the adversary's auxiliary information. The graph shows a significant reduction in the anonymity of the remaining hosts in the dataset, with over 50% of hosts experiencing a reduction in entropy of approximately 20 bits. Intuitively, the reason for this reduction is that the aux_t relation has been augmented with more accurate information about the anonymized prefixes. Consequently, the remaining hosts are placed into their correct subnets and the set of possible true identities is reduced substantially in some cases. This same approach can be used by data publishers to anticipate attacks, and examine various "what-if" scenarios in an objective and quantifiable manner.

Discussion and Future Work

The techniques provided in this paper can be computationally intensive, requiring significant time to fully compute the object anonymity and conditional object anonymity. Informally, our analysis is quadratic in the number of objects being analyzed, and when applied to the JHUISI dataset the analysis completed in approximately 8 hours on a single 2.4GHz processor.

One avenue of future work lies in the creation of methods for reducing the computational expense of our analysis while maintaining its correctness. We do note, however, that network data collection and anonymization is generally an offline process performed over the course of several weeks or months, and once published, the data may remain available to the public for several years. Consequently, the time spent evaluating the privacy implications of the anonymized network data is short relative to the other steps in the process, and this evaluation pays substantial dividends to the data publisher in the form of increased confidence in the sanitization of the data. Other important areas of future work include methods for efficiently examining a wider range of relationships within the data, and the application of our analysis technique to other object types, such as web pages.

2.8 Conclusion

Since the continued availability of anonymized network data relies heavily on the successful application of anonymization techniques—and the data publisher's confidence in the efficacy of those techniques—we believe that the analysis methods in this report provide several tangible benefits. Specifically, we present the first methods, of which we are aware, for analyzing the anonymity of network data. Using our analytical techniques, we show how data publishers can make informed decisions about the appropriate use of anonymization tools and the publication of anonymized data, thereby gaining confidence in the privacy of that data. We further demonstrate the utility of our techniques by showing their use in quantifying the anonymity of host objects, objectively comparing datasets and anonymization techniques, and examining the effects of deanonymization. It is our hope that the methodology presented in this report, along with the continued evolution of anonymization techniques and public data repositories, will further encourage the sharing of anonymized network data.

Chapter 3

The Challenges of Effectively Anonymizing Network Data

3.1 Summary

At first glance, it would seem as though the accumulated knowledge of microdata anonymization can be directly applied to network data anonymization since the two scenarios share so much in common, including similar privacy and utility goals. Unfortunately, the inherently complex nature of network data makes direct application of these microdata methods difficult, at best. We can, however, learn from existing microdata anonymization literature and glean significant insight into how to approach the problem of network data anonymization in a principled fashion.

In this exploration, we compare and contrast the fields of microdata and network data anonymization to reveal the ways in which existing microdata literature may be applied to the network data anonymization problem. We laid out several challenges that lie ahead in the development of robust network data anonymization methodologies that are grounded in the insights and lessons learned from microdata anonymization. Specifically, we examine the difficulties of clearly defining the privacy properties of network data due to its complex nature.

Our primary focus in drawing comparison between the fields of microdata and network data anonymization serves to focus the attention of the research community on a holistic approach to network data anonymization that enables the type of collaboration necessary to further progress in the areas of network and computer security research. The uncertainties that currently exist about the efficacy of network trace anonymization, from both technical and policy perspectives, leave the research community in a vulnerable position. Even as the field marches forward, it does so with little understanding of the implications of publishing anonymized network data on the privacy of the networks being monitored and the utility to researchers.

Without that understanding, data publishers are left to wonder what fields must be anonymized to avoid legal fallout, while researchers question the confidence of results gained from the data. However, the extensive work done on microdata anonymity provides the network research community with several useful insights about how to effectively apply anonymization to published data. At the same time, this prior wisdom cannot be applied directly without first overcoming several challenges, including the development of appropriate privacy and utility definitions for the more complex case of network data. **Addressing these challenges is essential to ensure the continued, yet**

responsible, availability of network trace data to support security research. Our subsequent work takes this advice to heart, and provides several new opportunities for pushing the field forward with respect to network trace anonymization.

3.2 Introduction

The availability of realistic network data plays a significant role in fostering collaboration and ensuring U.S. technical leadership in network security research. Unfortunately, a host of technical, legal, policy, and privacy issues limit the ability of operators to produce datasets for information security testing. In an effort to help overcome these limitations, several invaluable data collection efforts (e.g., CRAWDAD [26], `PREDICT` [66], and the Internet Measurement Data Catalog [78]) have been established in the past few years. The key principle used in all of these efforts to assure low-risk, high-value data is that of trace *anonymization*—the process of sanitizing data before release so that potentially sensitive information cannot be extracted.

Recently, however, the utility of these techniques in protecting host identities, user behaviors, network topologies, and security practices within enterprise networks has come under scrutiny. In short, several works have shown than unveiling sensitive data in anonymized network traces may not be as difficult as initially thought. The na¨*i*ve solution to this problem is to address the specifics of these attacks as they are discovered. However, doing so fails to address the underlying problem in its entirety. While isolated advances in network data anonymization are important, without a holistic approach to the problem they will simply shift the information-encoding burden to other properties of the traces, resulting in future privacy breaches. Given the significant reliance on anonymized network traces for security research, it is clear that a more exhaustive and principled analysis of the trace anonymization problem is in order.

Luckily, the problem of anonymizing publicly released data is not new. Over the past several decades, statisticians and computer scientists have developed approaches to anonymizing various forms of microdata, which are essentially databases of attributes collected about individuals. One prominent example is census data, which collects information about the salary, marital status, and many other potentially sensitive attributes from the population of an area or country. This census microdata, much like network data, is invaluable to researchers for tracking trends, and as such the anonymized microdata must provide accurate information about potentially sensitive information. At the same time, it is essential that specifics from the data can not be linked to individuals who participated in the survey. In response, several anonymization methods, privacy definitions, and utility metrics have been developed to ensure that researchers can use the microdata for a wide spectrum of analyses while simultaneously providing principled, concrete guarantees on the privacy of those individuals within the data.

At first glance, it would seem as though the accumulated knowledge of microdata anonymization can be directly applied to network data anonymization since the two scenarios share so much in common, including similar privacy and utility goals. Unfortunately, the inherently complex nature of network data makes direct application of these microdata methods difficult, at best. We can, however, learn from existing microdata anonymization literature and glean significant insight into how to approach the problem of network data anonymization in a principled fashion.

In this extended abstract, we compare and contrast the fields of microdata and network data anonymization to reveal the ways in which existing microdata literature may be applied to the

network data anonymization problem. We further lay out several challenges that lie ahead in the development of robust network data anonymization methodologies that are grounded in the insights and lessons learned from microdata anonymization. Specifically, we examine the difficulties of clearly defining the privacy properties of network data due to its complex nature. In addition, we point out the necessity of utility measures in quantifying the extent to which anonymization may alter results obtained from analysis of the data. It is important to note that there are additional challenges that we do not address herein, such as the legality of collecting network data [12, 60] and the ethical uses of such data [1]. As a whole, we hope that this comparison between the fields of microdata and network data anonymization serves to focus the attention of the research community on a holistic approach to network data anonymization that enables the type of collaboration necessary to further progress in the areas of network and computer security research.

3.3 Methods, Assumptions, and Procedures

Roughly speaking, microdata can be thought of as a database with n rows and m columns, where each row in the microdata corresponds to a single entity that contributed its data. In the case of census data, for example, the rows might represent people who responded to the survey. The columns represent the attributes of those entities, such as their height or salary information. Generally, these attributes come in two flavors: (*i*) categorical, or (*ii*) continuous (i.e., numeric). Continuous attributes encode values from a metric space that represents the semantics of the data, while categorical attributes do not have a natural ordering. As an example, salary information might be considered continuous in nature because the ordering of values is well-defined and agrees with the semantics of the data. Marital status or sex, on the other hand, have no ordering that make sense given the semantics of the data. These two attribute types lead to slight differences in the application of anonymization methods, though the underlying principles remain the same.

The goal of microdata anonymization is to alter the original data such that it is difficult to infer potentially sensitive information about entities within the data while simultaneously ensuring that statistics computed on the data remain unchanged. As an example, average salary information for a given area should remain unchanged, but it should not be possible to infer a specific person's salary. Specifically, the attributes are divided into three categories: (*i*) identifiers, (*ii*) key attributes (i.e., quasi-identifiers), and (*iii*) sensitive attributes. Identifiers are attributes that trivially identify the row, such as name or social security number. Key attributes can be used to make inferences on the identity of the row from auxiliary sources of information. Though these key attributes do not directly identify the row, unique attribute values can be used to link rows in the anonymized microdata with other databases that do have identifying information. For instance, if a row in the microdata had a unique combination of height, weight, and age key attributes, then the adversary could use these attributes to look up the row's identity in a secondary database that includes the height, weight, age, and name. Finally, sensitive attributes are those that are not available from other data sources, and which the adversary would like to link to specific identities. To achieve the goal of anonymization, the data publisher applies one or more anonymization methods to alter the relationship between the key attributes and sensitive attributes to ensure that such inferences are unlikely. The resultant sanitized microdata can then be measured to quantify its level of privacy and utility.

3.3.1 Anonymization Methods

Several techniques are used by data publishers to anonymize microdata for publication. Truncation methods remove or reorganize records in the microdata to hide the relationship between the key attributes and sensitive attributes. These methods include removing rows, removing attributes, suppression of key attribute values in specific rows, or generalization (i.e., recoding) where several key attributes are combined into a single equivalence class (e.g., $25 \leq age \leq 35$) [4, 75]. Additionally, several methods based on perturbation of the sensitive attributes exist. Some examples of perturbation include swapping the values of sensitive attributes among different rows [28], sampling the data, or adding noise to the values [7, 32]. At a high level, the truncation methods can be thought of as a method for adding uncertainty about the identity associated with the row, while perturbation methods adding uncertainty to the sensitive attribute that might be linked to the identity.

In addition to the truncation and perturbation-based methods, two methods have have been proposed which do not directly sanitize the microdata, but instead provide the underlying statistics of the data in alternate ways. The first of which, synthetic data generation [50, 68, 73], attempts to model the original data and generate completely new microdata from that statistical model. Since this new data is generated from a model, the resultant microdata has no connection to real individuals and at the same time the specific statistical properties captured by the model are guaranteed to be preserved. The second method stores the data on a secure remote server, where the data use can access it only through a query interface [6, 35, 76]. Thus, the user only gets the answer to specific queries, such as the average value of an attribute for a specific subgroup, and the query interface ensures that no queries are answered if they are harmful to privacy.

3.3.2 Measuring Privacy

Obviously, na¨*i*vely applying anonymization methods to the data is not enough to guarantee privacy. In fact, inappropriate application of anonymization methods may provide several avenues of information leakage. For instance, a recent study by Narayanan and Shmatikov [56] showed that an anonymized dataset of movie recommendations released by NetFlix fails to meet the accepted privacy definitions for microdata, which results in re-identification of several users in the data. To prevent such information leakage, it is necessary to concretely measure the privacy of the resultant anonymized data. As the extensive literature in microdata privacy measures indicates, however, developing privacy definitions that encapsulate all areas of information leakage are not as straightforward as one might hope.

The first of the microdata privacy definitions was proposed by Samarati and Sweeney [75], known as k-anonymity. The definition quantifies the difficulty of an adversary in determining which row in the microdata belongs to a given identity by requiring that every row must look like at least $k-1$ other rows with respect to their key attributes. In effect, this creates equivalence classes of key attributes where the adversary would have a $1/k$ chance of identifying the correct row using the key attributes. The k-anonymity definition, which uses generalization and suppression methods, was shown to be computationally infeasible to achieve deterministically, though approximation algorithms exist [52]. Chawla *et al.*provide a similar notion of anonymity that applies to purely continuous data, but which can be achieved with efficient anonymization methods based on perturbation [13].

The notion of k-anonymity provides a necessary, but not sufficient, condition for privacy since without it a row can be trivially identified by the uniqueness of its key attributes. Further restrictions are necessary, however, when we want to prevent the inference of sensitive attributes and not just which rows belong to a given identity. It may be possible, for example, to have an equivalence class that meets the k-anonymity definition, and yet has only one or a small number of distinct sensitive values. Thus, any individual that falls into such a class will have their sensitive attributes revealed. Machanavajjhala *et al.*[47] proposed ℓ-diversity to strengthen the k-anonymity property by requiring that each class have at least ℓ distinct sensitive values. Truta and Vinay [84] concurrently developed p-sensitive k-anonymity to provide the same requirement.

The ℓ-diversity property was further strengthened by Li *et al.*[45] since it may still be possible to leak information (in an information theoretic sense) about the sensitive attributes for an individual if the distribution of sensitive values in that individual's equivalence class are significantly different than those of the population. Essentially, the distribution within the equivalence class gives the adversary a more refined distribution of potential sensitive values for an individual than the adversary would have access to without the anonymized microdata. The t-closeness property [45] requires that the distribution of sensitive values in all equivalence classes be within a distance t of the population distribution across all rows in the microdata. This property ensures that the data publisher has greater control over the amount of information the adversary can gain about sensitive values of the individuals in the equivalence classes.

While k-anonymity and t-closeness provide controls over the information disclosed by the key attributes and sensitive attributes, respectively, there are still other avenues of information leakage which the adversary can take advantage of. Zhang *et al.*[90] showed that if the adversary has knowledge of the anonymization method used, she can reverse the anonymized key attributes to recover their original values. Specifically, during anonymization the original key attribute values are replaced by generalized versions (i.e., ranges of values) that attempt to create equivalence classes in such a way that equivalence class size is minimized. If the generalized value that was used for anonymization could have only been created by one or a small number of original key values, the adversary has gained significant information about the original key value by using nothing more than knowledge of the generalization scheme used. Zhang *et al.*suggest the notion of p-safe, p-optimal anonymization, where anonymized microdata produced to meet privacy definition p (e.g., k-anonymity) is considered safe if it has more than one original microdata that could have produced it. The microdata is considered optimal if that anonymization has the smallest equivalence classes possible given the privacy and safety criterion. In essence, this privacy definition provides controls over the information disclosed by the anonymization procedure itself. Hence, all three privacy definitions, k-anonymity, t-closeness, and p-safety, are necessary for truly private microdata.

An alternative approach to these uncertainty, or indistinguishability, definitions is provided by the notion of differential privacy [31]. Differential privacy is primarily applied to interactive query systems where users interact with the data via a secure query interface. The definition is born out of the observation that a person's privacy may be compromised even if they are not in the data being protected. For instance, if the adversary knew a person's salary relative to the average, then discovering the average salary from the data would violate that person's privacy even if they were not present. The notion of differential privacy states that the probability of a privacy breach occurring for a person is similar whether or not that person's information is contained in the data. The primary difference between differential privacy and the uncertainty-based definitions is that

differential privacy is unable to quantify exactly what sensitive information could be leaked by the data, and instead focuses on the slightly more general guarantee that no additional harm will be done by adding a record.

3.3.3 Measuring Utility

The primary motivation for publishing anonymized microdata is to provide some utility, such as the ability to calculate statistics on the attributes, to researchers who make use of the data. Clearly, the data would be useless if the privacy definitions above are achieved at the expense of utility. As a result, several utility measures have been developed to provide researchers with metrics that allow them to gauge the confidence they should have in the results gained by analysis of the anonymized data. Most utility measures for microdata focus on specific utilities that are meant to be preserved. Brickell and Shmatikov [10] evaluate the loss of utility with respect to specific machine learning tasks after microdata has been anonymized. Other types of utility measures include comparing various statistics on the anonymized and original data, such as means and variance, through the use of confidence intervals [38] or Z-scores [49]. The obvious problem with such approaches is that they can only anticipate a limited set of utilities and they provide no guidance about other uses of the data.

Recently, some global utility measures have been proposed to try and quantify a wide range of utilities in a single metric. These global measures, however, can be difficult to interpret and often times do not strongly predict the available utilities. Specifically, these measures are generally correlated with the extent to which utility is preserved, but they are unable to communicate to the researcher the exact extent to which a particular utility is affected by the anonymization. For instance, Karr *et al.*'s use of the Kullback-Leibler divergence between the anonymized and original data as a utility measure provides a broad idea of the extent to which the two distributions of attribute values are similar, but that value is not specific enough to predict the extent to which other measures, such as means or variances, differ. Woo *et al.*[86] also provide a global utility measure by using the performance of a binary statistical classifier in distinguishing the anonymized and original data to measure utility.

3.4 Network Data Anonymization

Network data can be viewed in much the same way as microdata; containing n rows each representing a single packet (or summarized network flow) and m columns representing the fields in the packet. These fields typically contain a variety of data types, including IP addresses, time stamps, and port numbers. Unlike microdata, however, the overwhelming majority of these data types are categorical in nature, where the ordering of the values has no semantic meaning. For example, while port numbers have an ordering by virtue of the fact that they are integers, the ordering of those numbers has no impact on their meaning and in fact they can be considered to operate as categories. Moreover, some fields, like IP addresses, have a highly complex hierarchical ordering structure that is not as straightforward to characterize as numerical data. Finally, the relationship among different fields in network data is semantically rich, which means that the values taken by certain fields is dependent on their context with respect to other values within the data – both within the same row and within other rows. Consider, for example, packet data for a TCP connection. The TCP flags, port numbers, and even packet sizes must follow certain restrictions based on

the network protocols being used, both within a single packet and across multiple packets in the connection to remain semantically meaningful.

The goals of network data anonymization are similar in nature to those of microdata insofar as they are focused on preventing the disclosure of sensitive information about certain entities present within the data. However, these goals are far more nebulous in the network data case since this sensitive information cannot be defined as a single field, nor can it be quantified for just a single row. Network data publishers are concerned with the privacy of workstations on the network and their users, which can be associated with multiple rows (i.e., packets) within the data. The sensitive information about these entities is often encoded in complex relationships among multiple fields across several different rows, such as a user's web browsing patterns or computer virus activity. Unfortunately, these goals remain ill-defined even in the most recent work in this area, which necessarily limits the efficacy of the anonymization procedures.

3.4.1 Anonymization Methods

Currently, the anonymization of network data is performed by applying one of a limited number of techniques, many of which are shared with microdata, to fields in the data chosen by the data published and defined in an anonymization policy language [64, 80]. The most widely used of these techniques are truncation, randomization, quantization, and pseudonymization. Truncation and randomization effectively destroy the semantics of the field they are applied to, but are helpful when dealing with fields that are likely to contain highly sensitive data. One example is the payload of packets, which might contain usernames and passwords and are removed from the data as standard practice. Quantization techniques, such as limiting the precision of time stamps, are applied to reduce the information gained about the identity of the workstations from timing attacks [40]. Finally, the most widely used technique focuses on replacing IP addresses found in the data with linkable, prefix-preserving pseudonyms [34, 63]. These pseudonyms preserve the hierarchical relationships found in the prefixes of the original addresses. The underlying goal is to enable the analysis of packets generated from hosts, or whole prefixes, without providing the actual IP. An alternative solution to providing anonymized network data directly to researchers was recently proposed by Mirkovic [53], which makes use of a secure query interface in a fashion similar to that of microdata. In this scenario, the user is only allowed to make a limited range of queries that are approved manually by the data publisher. Since the system is not built on top of sound privacy definitions, it must rely upon the data publisher to prohibit certain queries as new attacks are discovered.

In an effort to maintain as much of the original data as possible, data publishers apply these methods to as few fields as possible; normally, just the IP addresses, time stamps, and payloads. In fact, fields within the network data are typically anonymized only when they are shown to leak information via published attacks. As a result, the unaltered fields of the data provide significant information that can be used as key attributes to link objects in the data to their real identities. This reactionary anonymization policy has lead to the discovery of several attacks which use the unaltered features of the data to re-identify workstations and their behaviors [8, 9, 23, 70], and identify web pages that the users visit [19, 41].

3.4.2 Measuring Privacy

Given the reactionary nature of network data anonymization, it comes as no surprise that network data does not have well-defined privacy measures, due in part to the difficulty in clearly defining the privacy properties desired by data publishers. There have been a few attempts at quantifying the privacy of anonymized data in a fashion similar to k-anonymity. That is, to quantify the uncertainty that the adversary has in identifying which pseudonyms or values in the data belong to which real world workstations. Ribeiro *et al.*[70] derive fingerprints, such as port numbers used, for each IP address in both the anonymized and original data, and compare the two to determine the equivalence classes for each IP address. Those workstations with highly unique fingerprints are considered to be privacy risks for the data publisher. Coull *et al.*[24] also examines the similarity between the anonymized and original data, but examines a broader range of distributions of values found in the data rather than a limited set of fingerprints. In doing so, they quantify the privacy of workstations in terms of the number of other workstations with similar value distributions, and also discover those fields in the data that affect privacy the most. Kounine and Bezzi [42] perform a similar analysis with respect to the privacy of individual values after they have been anonymized rather than workstation privacy as a whole. The problem, of course, is that each of these techniques focus exclusively on workstation or individual field privacy, and yet network data can contain several different types of entities whose privacy is equally important. An alternate view of the data, for example, might be to look at it as a collection of web pages, and the privacy of a given web page is defined by the extent to which it is distinguishable from others in the data. There are many such views that one can take on the data, and this multifaceted property of network data makes it extremely difficult, if not impossible, to derive a single privacy metric that adequately captures all angles.

3.4.3 Measuring Utility

The idea of quantifying the utility of network data is only just beginning to gain traction in the network data anonymization community, though the complex nature of the data makes such measures as important, if not more so, as those proposed in microdata anonymization. One such utility measure was recently proposed by Lakkaraju and Slagell [44], and compares the performance of a well-known intrusion detection system on the anonymized and unanonymized data. Another measure was proposed by Burkhart *et al.*[11] and applies anomaly detection methodologies to the anonymized data to quantify the way in which it affects its performance. Both methods closely resemble those of Brickell and Shmatikov [10] that apply machine learning tasks to microdata to determine the degradation in accuracy, which can also be applied to network data. In addition, the global utility measure of Woo *et al.*[86] can also be applied with some adaptation to network data due to its use of standard statistical classification techniques. As with microdata, the use of highly specific measures, such as evaluations under specific anomaly detection methodologies or intrusion detection systems, leads to results that may not be applicable in a more general context. Similarly, global measures still remain difficult to interpret due to their disconnection from concrete utilities, and may in fact be even more difficult to apply effectively to network data because of its inherently complex and interdependent nature.

3.5 Challenges and Recommendations

Clearly, the problem of anonymizing microdata has received significant attention over the past three decades, and that attention has served to develop several methodologies for providing private and useful microdata to researchers. It is equally clear that network data anonymization is only just beginning to mature as an area of active research, and it can benefit from the substantial body of work generated by microdata anonymization research due to the similarities between the two areas. However, as discussed earlier, microdata and network data have a number of non-trivial differences that make direct application of well-known microdata anonymization concepts meaningless. In this section, we outline three broad challenges that lie ahead in the development of effective methods for anonymizing network data.

3.5.1 What are we protecting

Before we can begin developing effective anonymization methods for network data, we must first have a clear understanding of exactly what it is we hope to protect. For microdata, this question is easily answered because there is a natural one-to-one correspondence between the rows in the data and the entities being protected. With network data, however, this connection is not as clear. Publishers of network data are interested in protecting the privacy of a number of entities: the network users, the network's security procedures, and the hosts that operate on the network. What makes it difficult to clearly define these entities is the fact that network data is inherently multifaceted. A single record in the network data may actually affect the privacy of many entities of varying types. Moreover, the privacy of those entities is not contingent on only a single row in the data, but on many rows that define their behavior over time. While we could na¨ively apply existing microdata privacy definitions, these definitions would do nothing to ensure the privacy of the complex entities that are represented in network data. These issues naturally raise questions about how we define each of the entities for which the data publisher is interested in providing privacy.

With that said, for some types of entities the answer to this question is relatively straightforward. When considering the privacy of hosts on the network, for example, these host entities can be defined by assuming that the IP addresses in the network data consistently and uniquely identify a host. Ribeiro *et al.*[70] and Coull *et al.*[24] successfully use this definition of hosts in their privacy measurements. Even so, the relatively simple entity definition of hosts is not without its caveats, such as the possibility that multiple hosts may use the same IP. More complex entities, like users or web pages, are much more difficult to define without significant auxiliary information (e.g., audit logs). Using those auxiliary data sources to mark the entities associated with each record in the data is one potential avenue for defining the entities of interest in the network data.

3.5.2 What is sensitive

Network data has a wide variety of information encoded within it. One need only consider some of its uses in network research to appreciate its scope: e.g., measurements of network traffic characteristics, testing new networking methodologies and tools, and studying emerging phenomena. As we move forward, we must decide which of these pieces of information encoded within the network data should be considered to be sensitive. Again, the relatively simple structure of microdata allows

for an elegant definition of sensitive information – any attribute in the data that is likely to be unavailable from an external information source should be labeled as sensitive. The sensitivity of attributes are often easily intuited from knowledge of the underlying data. For instance, a data publisher can easily understand that information about a person's religion, disease status, or income level are potentially sensitive. Unfortunately, such intuitive definitions are simply not applicable to network data.

The very same information-rich properties that make network data so useful to the research community also lead to two significant challenges in defining which pieces of information might be considered to be sensitive. First, potentially sensitive information encoded within the network data is not restricted to a single column in the data. In fact, the relationships between the columns and across several records often indicate the most sensitive of information. For instance, it may not be possible to determine if a given host within the network data is infected by a computer virus by simply examining the distribution of ports it has used, but that distribution of ports in combination with other fields may very well reveal its infection status. Similar arguments could be made for whether a user visited an illicit web site, or if the network is using a particular security system. Second, many of the fields present within network data contain values that an adversary would be able to recover from external sources, as well as values that are not possible to find elsewhere. As an example, the distribution of ports used by a host may indicate the services it offers. For an adversary with access to certain public information, some of the ports might be public knowledge while others, such as those offered privately within the network may be unknown from the public records. These scenarios are particularly troublesome since the known values within the column of port numbers can act as key attributes, while the unknown values act as sensitive attributes that the adversary may seek to infer.

Many of the attacks that have been discovered for anonymized network data take advantage of these issues in subverting the privacy of the data. Host profiling attacks [8, 9, 23, 70], for instance, use some of the ports and IP pseudonyms in the data as key attributes to link the hosts to their real identities, and then use the remaining ports to infer the hosts hidden services. Web page identification attacks [19, 41], on the other hand, use flow and packet sizes to infer the identity of the web pages being viewed by users on the network – the sensitive attribute here is the mere fact that a user happened to be associated with a set of sizes that indicate a given web page. Rather than attempt to adapt the static notions of key and sensitive attributes to multifaceted network data, current approaches to measuring privacy of network data (e.g., [24, 42, 70]) instead focus on the uniqueness of a piece of data as an indicator for sensitivity. The underlying assumption is that a sufficiently unique behavior encoded within the data is likely to be unavailable from other data sources. Other definitions of sensitivity may be possible based on the specific privacy concerns of the publisher, though relying on intuitive notions of sensitivity in this case will likely lead to unintended information leakage.

3.5.3 Defining Utility for Network Data

An area of considerable interest for both microdata and network data anonymization is the development of metrics that measure the utility of the data after it has been anonymized. These metrics are especially important in the case of network data, where the inherent difficulties of defining sensitivity and entities within the data may lead to essentially useless data. For instance, if we follow the definition that sensitive information in network data is any piece of information that

is sufficiently unique, then it is easy to imagine a scenario in which the network data contains only homogenous behavior. This type of homogenous data would be useless to researchers who are interested in investigating anomalous incidents or who want to get an accurate estimation of traffic characteristics. In these types of scenarios, it is imperative that researchers have access to utility metrics with respect to certain properties of the data so that they, and those that review their work, can adequately gauge its appropriateness to the task at hand.

Specific utility measures may provide an adequate short term solution to the problem. In general, a utility measure can be derived by comparing the results of a particular utility on the anonymized data to those of the unanonymized data. The problem, of course, lies in predicting the utilities that will be used. One simple way to address this concern is for the data publisher to publish a set of metrics for standard utilities on the data, and allow researchers to request additional utility measures as necessary. However, this type of arrangement is a significant burden on data publishers and researchers, since data publishers would need to run various analyses on the data and researchers would be unable to perform exploratory analyses in a timely fashion. A slightly different approach might be to adapt the concept of a remote access server, like those used in microdata [6, 35, 76] and network data [53], to allow researchers to automatically verify the results of their analyses on the anonymized data. That is, the researcher can send her results for a specified query to the remote server, and the remote server can automatically compute the query on the original data then return a utility metric to the researcher. A recent proposal by Reiter *et al.*[69] proposes just such a verification service for statistics computed on microdata. A slightly more elegant approach would be one in which the privacy and utility of the data were summarized in a single metric, thereby following the intuition that increased privacy necessarily decreases utility on a broad scale and vice-versa.

3.6 Conclusion

The uncertainties that currently exist about the efficacy of network trace anonymization, from both technical and policy perspectives, leave the research community in a vulnerable position. Even as the field marches forward, it does so with little understanding of the implications of publishing anonymized network data on the privacy of the networks being monitored and the utility to researchers. Without that understanding, data publishers are left to wonder what fields must be anonymized to avoid legal fallout, while researchers question the confidence of results gained from the data. However, the extensive work done on microdata anonymity provides the network research community with several useful insights about how to effectively apply anonymization to published data. At the same time, this prior wisdom cannot be applied directly without first overcoming several challenges, including the development of appropriate privacy and utility definitions for the more complex case of network data. Addressing these challenges is essential, in our view, to ensure the continued, yet responsible, availability of network trace data to support security research.

Chapter 4

On Measuring the Similarity of Network Hosts

4.1 Summary

Network operators face a bewildering array of security and operational challenges that require significant instrumentation and measurement of their networks, including denial of service attacks, supporting quality of service, and capacity planning. Unfortunately, the complexity of modern networks often make manual examination of the data provided by such instrumentation impractical. As a result, operators and security practitioners turn to automated analysis methods to pinpoint interesting or anomalous network activities. Underlying each of these methods and their associated applications is a fundamental question: to what extent are two sets of network activities similar?

As straightforward as this may seem, the techniques for determining this similarity are as varied as the number of domains they have been applied to. These range from identifying duplicates and minor variations using cryptographic hashes and edit distance of payloads, to the use of distributions of features and their related statistics (e.g., entropy). While each has been shown to provide value in solving specific problems, the diversity of approaches clearly indicates that there is no generally accepted method for reasoning about the similarity of network activities.

With this in mind, *our goal in this part of the project was to make progress in developing a unified approach to measuring the similarity of network activities.* In doing so, we hoped to encourage a more rigorous method for describing network behaviors, which will hopefully lead to new applications that would be difficult to achieve otherwise. While a complete framework for rigorously defining distance is beyond the scope of any one piece of work, we address two key aspects of network similarity that we believe must be considered in any such a framework. That is, the *spatial* and *temporal* characteristics of the network data.

Here, *spatial* characteristics refer to the unique semantic relationships between the values in two identical or related fields. For example, if we wish to cluster tuples of data that included IP addresses and port numbers, we would have two obvious ways of accomplishing the task: (1) treat the IPs and ports as numeric values (i.e., integers) and use subtraction to calculate their distance, or (2) treat them as discrete values with no relationship to one another (e.g., in a probability distribution). Clearly, neither of these two options is exactly correct, since the network protocols that define these data types also define unique semantic relationships.

Temporal characteristics, on the other hand, describe the causal relationships among the network activities over time. One way to capture temporal information is to examine short n-grams or k-tuples of network activities. However, this too may ignore important aspects of network data. As an example, changes in traffic volume may alter the temporal locality captured by these short windows of activity. Moreover, network data has no restriction on this temporal locality, which means that activities can have long-term causal relationships with each other, extending to minutes, hours, or even days. Successfully building robust notions of similarity that address these temporal characteristics mean addressing similarity over large time scales.

In this aspect of the project, we explored the spatial and temporal properties in an effort to learn what impact they may have on the performance of automated network analysis methods. To focus our study, we examine the problem of classifying host behaviors, which requires both strong notions of semantically-meaningful behavior and causal relationships among these behaviors to provide a meaningful classification. Furthermore, we develop an example unified metric space for network data that encodes the unique spatial and temporal properties of host behavior, and evaluate our proposed metric by comparing it to one that ignores those properties. We note that it is not our intention to state that current analysis methodologies are "wrong," nor that we present the best metric for computing similarity. Instead, we explore whether general metrics for network activities can be created, how such metrics might lead to improved analysis, and examine the potential for new directions of research.

More concretely, we begin by defining metric spaces for a subset of data types commonly found within network data. We design these metric spaces to encode the underlying semantic relationship among the values for the given data type, including non-numeric types like IP addresses and port numbers. Then, we show how to combine these heterogeneous metric spaces into a unified metric space that treats network data records (e.g., packets, network flows) as points in a multi-dimensional space. Finally, we describe host behaviors as a time series of these points, and provide a dynamic time warping algorithm that efficiently measures behavioral distance between hosts, even when those time series contain millions of points. In doing so, we develop a geometric interpretation of host behavior that is grounded in semantically-meaningful measures of behavior, and the long-term temporal characteristics of those behaviors. Wherever possible, we brought to light several challenges that arise in the development of general metrics for network data.

To evaluate our proposed metric, we used a dataset containing over 30 million flows collected at the University of Michigan. Our experiments compared the performance of our metric to that of the L_1 (i.e., Manhattan distance) metric, which ignores semantics and temporal information, in a variety of cluster analysis tasks. **The results of these experiments indicated that semantic and temporal information play an important role in capturing a realistic and intuitive notion of network behaviors**. Furthermore, these results imply that it may be possible to treat network data in a more rigorous way, similar to traditional forms of numeric data. **To underscore this potential, we showed how our metric may be used to measure the privacy provided by anonymized network data in the context of well-established privacy definitions for real-valued, numeric data; namely, Chawla *et al.*'s (c, t)-isolation definition [13]**.

4.2 Introduction

Network operators face a bewildering array of security and operational challenges that require significant instrumentation and measurement of their networks, including denial of service attacks, supporting quality of service, and capacity planning. Unfortunately, the complexity of modern networks often make manual examination of the data provided by such instrumentation impractical. As a result, operators and security practitioners turn to automated analysis methods to pinpoint interesting or anomalous network activities. Underlying each of these methods and their associated applications is a fundamental question: to what extent are two sets of network activities similar?

As straightforward as this may seem, the techniques for determining this similarity are as varied as the number of domains they have been applied to. These range from identifying duplicates and minor variations using cryptographic hashes and edit distance of payloads, to the use of distributions of features and their related statistics (*e.g.,* entropy). While each has been shown to provide value in solving specific problems, the diversity of approaches clearly indicates that there is no generally accepted method for reasoning about the similarity of network activities.

With this in mind, the goal of this paper is to make progress in developing a unified approach to measuring the similarity of network activities. In doing so, we hope to encourage a more rigorous method for describing network behaviors, which will hopefully lead to new applications that would be difficult to achieve otherwise. While a complete framework for rigorously defining distance is beyond the scope of any one paper, we address two key aspects of network similarity that we believe must be considered in any such a framework. That is, the *spatial* and *temporal* characteristics of the network data.

Here, *spatial* characteristics refer to the unique semantic relationships between the values in two identical or related fields. For example, if we wish to cluster tuples of data that included IP addresses and port numbers, we would have two obvious ways of accomplishing the task: (1) treat the IPs and ports as numeric values (*i.e.,* integers) and use subtraction to calculate their distance, or (2) treat them as discrete values with no relationship to one another (*e.g.,* in a probability distribution). Clearly, neither of these two options is exactly correct, since the network protocols that define these data types also define unique semantic relationships.

Temporal characteristics, on the other hand, describe the causal relationships among the network activities over time. One way to capture temporal information is to examine short n-grams or k-tuples of network activities. However, this too may ignore important aspects of network data. As an example, changes in traffic volume may alter the temporal locality captured by these short windows of activity. Moreover, network data has no restriction on this temporal locality, which means that activities can have long-term causal relationships with each other, extending to minutes, hours, or even days. Successfully building robust notions of similarity that address these temporal characteristics mean addressing similarity over large time scales.

Contributions. In this paper, we explore these spatial and temporal properties in an effort to learn what impact they may have on the performance of automated network analysis methods. To focus our study, we examine the problem of classifying host behaviors, which requires both strong notions of semantically-meaningful behavior and causal relationships among these behaviors to provide a meaningful classification. Furthermore, we develop an example unified metric space for network data that encodes the unique spatial and temporal properties of host behavior, and evaluate our proposed metric by comparing it to one that ignores those properties. We note that it

is not our intention to state that current analysis methodologies are "wrong," nor that we present the best metric for computing similarity. Instead, we explore whether general metrics for network activities can be created, how such metrics might lead to improved analysis, and examine the potential for new directions of research.

More concretely, we begin by defining metric spaces for a subset of data types commonly found within network data. We design these metric spaces to encode the underlying semantic relationship among the values for the given data type, including non-numeric types like IP addresses and port numbers. Then, we show how to combine these heterogeneous metric spaces into a unified metric space that treats network data records (*e.g.,* packets, network flows) as points in a multi-dimensional space. Finally, we describe host behaviors as a time series of these points, and provide a dynamic time warping algorithm that efficiently measures behavioral distance between hosts, even when those time series contain millions of points. In doing so, we develop a geometric interpretation of host behavior that is grounded in semantically-meaningful measures of behavior, and the long-term temporal characteristics of those behaviors. Wherever possible, we bring to light several challenges that arise in the development of general metrics for network data.

To evaluate our proposed metric, we use a dataset containing over 30 million flows collected at the University of Michigan. Our experiments compare the performance of our metric to that of the L_1 (*i.e.,* Manhattan distance) metric, which ignores semantics and temporal information, in a variety of cluster analysis tasks. The results of these experiments indicate that semantic and temporal information play an important role in capturing a realistic and intuitive notion of network behaviors. Furthermore, these results imply that it may be possible to treat network data in a more rigorous way, similar to traditional forms of numeric data. To underscore this potential, we show how our metric may be used to measure the privacy provided by anonymized network data in the context of well-established privacy definitions for real-valued, numeric data; namely, Chawla *et al.*'s (c, t)-isolation definition [13].

4.3 Methods, Assumptions, and Procedures

For ease of exposition, we first provide definitions and notation that describe the network data we analyze in a format-independent manner. We also define the concepts of metric spaces and product metrics, which we use to create a foundation for measuring similarity among network hosts.

Network Data Data describing computer network activities may come in many different forms, including packet traces, flow logs, and web proxy logs. Rather than describe each of the possible formats individually, we instead define network data as a whole in more abstract terms. Specifically, we consider all network data to be a database of m rows and n columns, which we represent as an $m \times n$ matrix. The rows represent individual records, such as packets or flows, with n fields, which may include source and destination IP addresses, port numbers, and time stamps. We denote the i^{th} row as the n-dimensional vector $\vec{\mathbf{v}}_i =< v_{i,1}, \ldots, v_{i,n} >$, and the database as $V =< \vec{\mathbf{v}}_1, \ldots, \vec{\mathbf{v}}_m >^T$. For our purposes, we assume a total ordering on the rows $\vec{\mathbf{v}}_i \leq \vec{\mathbf{v}}_{i+1}$ based on when the record was added to the database by the network sensor. Furthermore, we associate each column in the matrix (*i.e.,* field) with an atomic data type that defines the semantic relationship among its values. More formally, we define a set of types $T = \{t_1, \ldots, t_\ell\}$ and an injective function $F : [1, n] \to T$ that maps a column to its associated data type.

Metric Spaces To capture a notion of similarity among values in each column, we define a metric space for each data type in the set T. A *metric space* is simply a tuple (X, d), where X is a non-empty set of values being measured and $d : X \times X \to \mathbb{R}_+$ is a non-negative distance metric. The metric function must satisfy three properties: (1) $d(x, y) = 0$ iff $x = y$; (2) $d(x, y) = d(y, x)$; and (3) $d(x, y) + d(y, z) \geq d(x, z)$. We denote the metric for the type t_j as (X_{t_j}, d_{t_j}).

Given the metric spaces associated with each of the columns via the data type mapping described above, $(X_{F(1)}, d_{F(n)}), \ldots, (X_{F(n)}, d_{F(n)})$, we can define a *p-product metric* to combine the heterogeneous metric spaces into a single metric space that measures similarity among records as if they were points in an n-dimensional space. Specifically, the p-product metric is defined as $(X_{F(1)} \times \ldots \times X_{F(n)}, d_p)$, where $X_{F(1)} \times \cdots \times X_{F(n)}$ denotes the Cartesian product of the sets and d_p is the p-norm:

$$d_p(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = (d_{F(1)}(x_1, y_1)^p + \cdots + d_{F(n)}(x_n, y_n)^p)^{\frac{1}{p}}$$

We note that metrics we propose are straightforward generalizations of well-known metrics [54]; hence, the proofs are omitted for brevity.

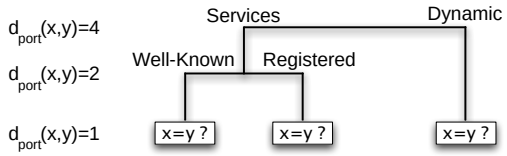
4.3.1 Metric Spaces for Network Data

To provide a foundation for measuring similarity among network hosts, we define a metric space that captures both the spatial and temporal characteristics of the host's behaviors as follows. We begin by defining metrics spaces that capture semantically rich relationships among the values for each data type found in the network data. For the purposes of this initial study, we restrict ourselves to providing example metrics for four prevalent data types: IP addresses, port numbers, time fields, and size fields. Next, we show how to combine these heterogeneous metric spaces into a single, unified metric space using a p-product metric and a novel normalization procedure that retains the semantic relationships of the constituent metric spaces. This allows us to treat each network data record as a point and capture the spatial characteristics of the host's behavior in a meaningful way. Finally, we model a host's temporal behavior as a time series of points made up of records associated with the host (*e.g.,* flows sent or received by the host), and show how dynamic time warping may be used to efficiently measure distance between the behavior of two hosts.

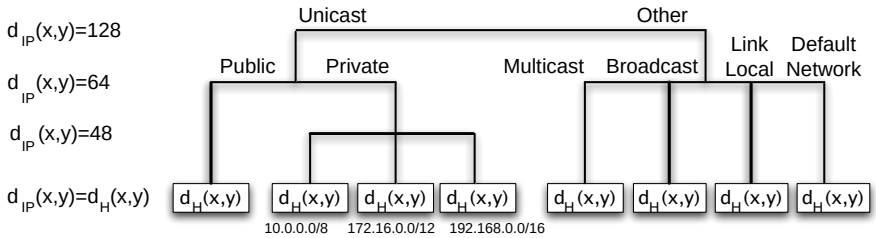
4.3.2 Data Types and Metric Spaces

Network data may contain a wide variety of data types, each with its own unique semantics and range of possible values. That said, without loss of generality, we can classify these types as being numeric (*e.g.,* timestamps, TTL values, sizes) or categorical (*e.g.,* TCP flags, IPs, ports) in nature. The primary distinction between these two categories of types is that numeric types have distance metrics that naturally follow from the integer or real number values used to represent them, while categorical types often do not maintain obvious linear relationships among the values. Here, we describe example metric spaces for two data types in each category: time and size as numeric types, and IP and port as categorical.

Numeric Types. The time and size data types contain values syntactically encoded as 16 or 32 bit integers, and have semantics that mirror those of the integers themselves. That is, a distance



(a) Distance hierarchy for port type.



(b) Distance hierarchy for IP type.

Figure 4.1: Distance metrics for categorical types of port and IP. Distances listed to the left indicate the distance when values diverge at that level of the hierarchy.

of ten in the integers is semantically the same as ten bytes or ten seconds[1]. Both types can be represented by metric spaces of the form $(X = \{0, \ldots, M\}, d(x, y) = |x - y|)$, where M is simply the largest value possible in the encoding for that type. In most cases, that means that $M = 2^{32} - 1$ for 32-bit integers, or $M = 2^{16} - 1$ for 16-bit.

An obvious caveat, however, is that one must ensure that values are encoded in such a way that they are relevant to the analysis at hand. For example, when comparing host behavior over consecutive days, it makes sense to ensure time fields are made relative to midnight of the current day, thereby allowing one to measure differences in activity relative to that time scale (*i.e.,* a day) rather than to a fixed epoch (*e.g.,* UNIX timestamps).

Categorical Types. While metric spaces for numeric types are straightforward, categorical data types pose a problem in developing metric spaces due to the non-linear relationship between the syntactic encoding (*i.e.,* integers) and the underlying semantic meaning of the values. Moreover, the unique semantics of each data type prohibits us from creating a single metric for use by all categorical types. Instead, we follow a general procedure whereby a hierarchy is defined to represent the relationships among the values, and the distance between two values is determined by the level of the hierarchy at which the values diverge. While this is by no means the only approach for defining metrics for categorical types, we believe it is a reasonable first step in representing the semantics of these complex types.

To define the distance hierarchy for the port type, we decompose the space of values (*i.e.,* port numbers) into groups based on the IANA port listings: well-known (0-1023), registered (1024-49151), and dynamic (49151-65535) ports. In this hierarchy, well-known and registered ports are considered to be closer to one another than to dynamic ports since these ports are typically statically associated with a given service. Of course, ports within the same group are closer than those in different groups, and the same port has a distance of zero. The hierarchy is shown in Figure 4.1(a).

[1]Note that we will address issues arising from different units of measure in the following section.

More formally, we create the metric space (X_{port}, d_{port}), where $X_{port} = \{0, \ldots, 65535\}$ and d_{port} is an extension of the discrete metric defined as:

$$d_{port}(x,y) = \begin{cases} 0 & if\ x = y \\ 1 & if\ \delta_{port}(x) = \delta_{port}(y) \\ 2 & if\ \delta_{port}(x) \in \{0,1\}\ \&\ \delta_{port}(y) \in \{0,1\} \\ 4 & if\ \delta_{port}(x) \in \{0,1\}\ \&\ \delta_{port}(y) \in \{2\} \end{cases}$$

$$\delta_{port}(x) = \begin{cases} 0 & if\ x \in [0, 1023] \\ 1 & if\ x \in [1024, 49151] \\ 2 & if\ x \in [49152, 65535] \end{cases}$$

The choice of distances in d_{port} is not absolute, but must follow the metric properties from Section 4.3 and also faithfully encode the relative importance of the differences in groups that the ports belong to. One can also envision extending the hierarchy to include finer grained information about the applications or operating systems typically associated with the port numbers. Many such refinements are possible, but it is important to note that they make certain assumptions on the data which may not hold in all cases. For example, one could further refine the distance measure to ensure that ports 80 and 443 are near to one another as both are typically used for HTTP traffic, however there are clearly cases where other ports carry HTTP traffic or where ports 80 and 443 carry non-HTTP traffic. One of the benefits of this approach to similarity measurement is that it requires the analyst to carefully consider these assumptions when defining metrics.

The metric space for IP addresses[2] is slightly more complex due to the variety of address types used in practice. There is, for instance, a distinction between routable and non-routable, private and public, broadcast and multicast, and many others. In Figure 4.1(b), we show the hierarchy used to represent the semantic relationships among these groupings. At the leaves of this hierarchy, we perform a simple Hamming distance calculation (denoted as d_H) between the IPs to determine distance within the same functional group. Due to the complexity of the hierarchy, we do not formally define its metric space here. We again reiterate that this is just one of potentially many ways to create a metric for functional similarity of IP addresses.

4.3.3 Network Data Records as Points

Given the aforementioned distance metrics, the next challenge is to find a way to combine them into a unified metric space that captures the distance between records (*i.e.,* flows or packets) by treating them as points within that space. At first glance, doing so would appear to be a relatively simple procedure: assign one of the available types to each field in the record, and then combine the respective metric spaces using a p-product metric. However, when combining heterogeneous spaces it is important to normalize the distances to ensure that one dimension does not dominate the overall distance calculation by virtue of its relatively larger distance values. Typically, this is accomplished with a procedure known as affine normalization. Given a distance metric d on values $x, y \in X$, the affine normalization is calculated as:

$$\overline{d(x,y)} = \frac{d(x,y) - min(X)}{max(X) - min(X)}$$

[2]The provided metric is for IPv4 addresses. IPv6 addresses would simply require a suitable increase in distances for each level to retain their relative severity.

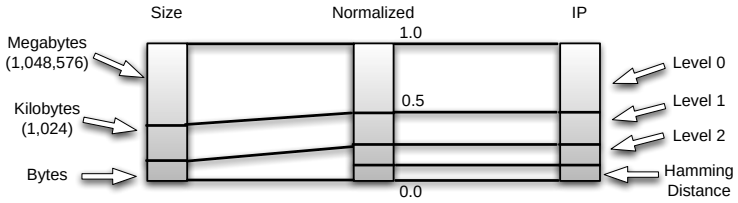


Figure 4.2: Example piecewise normalization of size and IP types. Mapping each range independently ensures they are weighted in accordance with the relative severity of the distance.

However, a naive application of this normalization method fails to fairly weight all of the dimensions used in the overall distance calculation. In particular, affine normalization ignores the distribution of values in the underlying space by normalizing by the largest distance possible. As a result, very large and sparse spaces, such as IPs or sizes, may actually be *undervalued* as a result.

To see why, consider a field containing flow size data. In the evaluation that follows, the vast majority of flows have less than 100 bytes transferred, but the maximum seen is well over ten gigabytes in size. As a result, affine normalization would give extremely small distances to the vast majority of flow pairs even though those distances may actually still be semantically meaningful. In essence, affine normalization procedures remove or minimize the semantic information of distances that are not on the same scale as the maximum distance. Yet another approach might be to measure distance as the difference in rank between the values (*i.e.,* the indices of the values in sorted order). Doing so, however, will remove all semantic information about the relative severity of the difference in those values.

To balance these two extremes, we propose a new procedure that normalizes the data according to common units of measure for the data type. We begin by defining the overall range of distances that are possible given the values seen in the data being analyzed. Then, we divide this space into non-overlapping ranges based upon the unit of measure that most appropriately suits the values in the range. In other words, the range associated with a given unit of measure will contain all distances that have a value of at least one in that unit of measure, but less than one in the next largest unit. In this paper, we use seconds, minutes, and hours for the time data type, and bytes, kilobytes, and megabytes for size types. Categorical types, like IPs and ports, are assigned units of measure for each level in their respective distance hierarchies.

Once each distance range is associated with a unit of measure, we can then independently map them into a common (normalized) interval such that the normalized distances represent the relative severity of each unit of measure with respect to units from the same type, as well as those from other data types that are being normalized. It is this piecewise mapping to the normalized distance range that allows us to maintain the semantic meaning of the underlying distances without unduly biasing certain dimensions in the overall distance calculation. For simplicity, we map all metric spaces with three units of measure to the ranges $[0, 0.25)$, $[0.25, 0.5)$, and $[0.5, 1.0]$. Types with four units are mapped as $[0, 0.125)$, $[0.125, 0.25)$, $[0.25, 0.5)$,$[0.5, 1.0]$. Figure 4.2 shows how this mapping is achieved for size and IP types. Then, by denoting the function that produces the normalized distance for type t_j as $\overline{d_{t_j}(x, y)}$, we can use the p-product metric to define the distance between

records in the network data as:

$$d_p(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = \left(\overline{d_{F(1)}(x_1, y_1)}^p + \cdots + \overline{d_{F(n)}(x_n, y_n)}^p\right)^{\frac{1}{p}}$$

The metric space is now $(X_{F(1)} \times \cdots \times X_{F(n)}, d_p)$, and we set $p = 2$ in order to calculate the Euclidean distance between records.

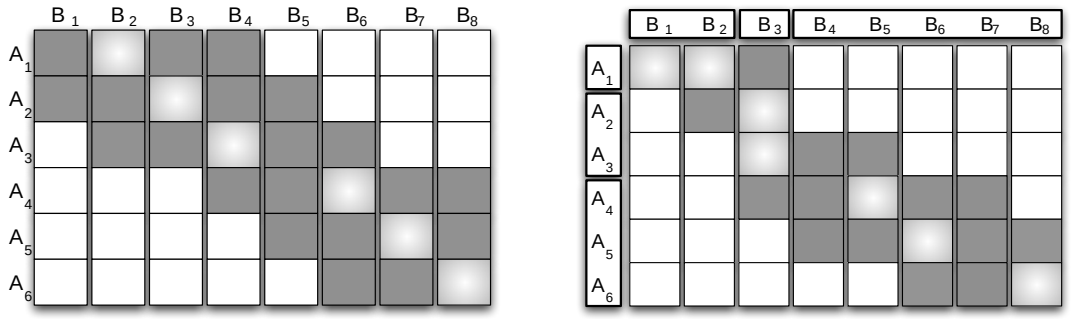
4.3.4 Network Objects as Time Series

Thus far, we have introduced distance metrics that attempt to capture the semantics that underlie various data types found in network data, and showed how that semantic information may be propagated to metrics for network data records by carefully normalizing and creating a unified metric space. The final step in our study requires us to take these records and show how to use the metrics we have developed to capture the similarity in behavior between two hosts. In effect, we defined a method for reasoning about the spatial similarity of individual records associated with a host, but must also address the concept of temporal similarity to capture its behavior in a meaningful way.

Common wisdom suggests that host similarity should be calculated by independently evaluating records associated with that host, or only examining short subsequences of activities. This is due primarily to the so-called "curse of dimensionality" and the sheer volume of behavioral information associated with each host. Our assertion is that doing so may be insufficient when trying to capture a host's activities and behaviors as a whole, since these behaviors often have strong causal relationships that extend far beyond limited n-gram windows.

In order to capture the entirety of a host's behavior and appropriately measure similarity among hosts, we instead model a host's behavior as a time series of points made up of the records embedded into the n-dimensional metric space described in the previous section. Given two hosts represented by their respective time series, a dynamic programming method known as *dynamic time warping* (DTW) may be used to find an order-preserving matching between points in the two series which minimizes the overall distance. The DTW approach has been used for decades in the speech recognition community for aligning words that may be spoken at different speeds. Essentially, the DTW procedure "speeds up" or "slows down" the time series to determine which points should be aligned based on the distances between those two points, which in our case are calculated using the p-product metric.

Unfortunately, DTW runs in $O(m_1 m_2)$ time and space for two time series of length m_1 and m_2, respectively. Considering that most real-world hosts produce millions of records per day, such an approach would quickly become impractical. Luckily, several heuristics have been proposed that limit the area of the dynamic programming matrix being evaluated. In particular, Sakoe and Chiba [74] provide a technique which reduces the computational requirements of the DTW process by evaluating only those cells of the matrix which lie within a small window around the diagonal of the matrix. Despite only examining a small fraction of cells, their approach has been shown to produce accurate results in measuring similarity due to the fact that most high-quality matches often occur in a small window around the diagonal, and that it prevents so-called pathological warpings wherein a single point may map to large sections of the opposite time series.



(a) DTW with heuristic for constant sampling.

(b) DTW with heuristic for variable sampling.

Figure 4.3: Example dynamic programming matrices with the original Sakoe-Chiba heuristic (a) and our adapted heuristic (b) for variable sampling rate time series. Gradient cells indicate the "diagonals" and dark shaded cells indicate the window around the diagonal that are evaluated. In the the variable rate example, points in the same subsequence are grouped together.

4.3.5 Efficient Dynamic Time Warping for Network Data

Ideally, we would apply the Sakoe-Chiba heuristic to make the DTW computation feasible even for very large datasets. Unfortunately, the Sakoe-Chiba heuristic makes assumptions on the time series that prevent its direct application to network data. In particular, it assumes that the points in the time series are sampled at a constant rate, and so the slope of the diagonal that is evaluated in the matrix depends only on the length of the time series (*i.e.,* the time taken to speak the word). For network data, however, the rate at which records are produced is based on the traffic volumes sent and received by the host, which are inherently bursty in nature and dependent on the host's activities. Consequently, a single diagonal with a fixed slope would yield a warping path that attempts to align points that may be recorded at significantly different times, leading to a meaningless measurement of behavior.

To address this, we propose an adaptation where we break the two time series into subsequences based on the time period those sequences cover, and calculate individual diagonals and slopes for each subsequence. The individual subsequences can then be pieced together to form a warping path that accommodates the variable sampling rate of network data, and from which we can extend a window to appropriately measure similarity without evaluating the entire dynamic programming matrix. Figure 4.3 shows an example of the traditional Sakoe-Chiba heuristic and our adaptation for network data.

Our extension of the Sakoe-Chiba heuristic for network data proceeds as follows. Assume that we are given two time series A and B, where $|A| \leq |B|$. We begin by splitting the two time series being aligned into subsequences such that all points in the same subsequence were recorded during the same second (according to their timestamps). The subsequences in the two time series are paired if they contain points for the same second. Any subsequences that have not been paired are merged in with the subsequence in the same time series that is closest to its timestamp. Thus, we have k subsequences $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$ for the sequences A and B, with A_i and B_i being mapped to one another. Next, we iterate through each of the k subsequence pairs and calculate the slope of the pair as $S_i = \frac{|B_i|}{|A_i|}$. The slope and subsequence mapping uniquely determine which

Algorithm 1
Dynamic Time Warp $(A, B, map, slope, c)$

Require: $|A| \leq |B|$
Require: $c > 0$
  Initialize m as a $|A| \times |B|$ matrix
  $m[i][j] \leftarrow \infty$ for all i, j
  $m[0][0] \leftarrow 0.0$
  for $i \leftarrow 0$ to $|A|$ **do**
    $start \leftarrow max(0, map[i] - slope[i] * c)$
    $end \leftarrow min(|B|, map[i] + slope[i] * c)$
    for $j \leftarrow start$ to end **do**
      if $i \neq 0$ and $j \neq 0$ **then**
        $distance \leftarrow d_p(A[i-1], B[j-1])$
        $left \leftarrow matrix[i][j-1] + distance$
        $up \leftarrow matrix[i-1][j] + distance$
        $diag \leftarrow matrix[i-1][j-1] + distance$
        $m[i][j] \leftarrow min(left, up, diag))$
      end if
    end for
  end for
  return $m[|A|][|B|]/(|A| + |B|)$

cells should be evaluated as the "diagonal" in our variable sampling rate adaptation. Specifically, the j^{th} point in the i^{th} subsequence (*i.e.,* $A_{i,j}$) is mapped to $B_{i,j'}$ where $j' = \lceil j * S_i \rceil$.

With this mapping among points in hand, we then place a window around each cell of length at least $\lceil S_i \rceil$ to ensure continuity of the path in the dynamic programming matrix (*i.e.,* a transition exists between cells). For those points that occur at the beginning or end of a subsequence, the window length is set based on the index of the mappings for the adjacent points to ensure that the subsequences are connected to one another. In order to provide a tunable trade-off between computational complexity and accuracy of the warping, we extend the window by a multiplicative parameter c so that $c * S_i$ cells on either side of the "diagonal" are evaluated.

Algorithm 1 shows the dynamic time warping procedure using the restricted warping path from our variable sampling rate heuristic[3]. For ease of presentation, we provide as input to the algorithm lists containing the two time series A and B, the mapping between indices in A and their "diagonal" mapping in B, a list containing the slope values to be used to ensure continuity of the matrix *slope*[4], and the multiplicative factor c that controls the number of cells evaluated around the "diagonals." As with any dynamic programming technique, the minimum distance for the allowable warping path is given in the lower-right cell of the matrix. Furthermore, in order to ensure the computed warping distances are comparable among time series of varying lengths, we normalize the distance by dividing by the number of cells in the longest possible warp path $|A| + |B|$, which also ensures that the distance remains symmetric. We use this normalized distance to compare the behaviors of hosts and determine their similarity. The computational complexity of our proposed heuristic is $O(|B| * c)$ in the worst case where both series A and B are of equal length, which represents a

[3]Note that this algorithm can be easily altered so that only two rows in the matrix are maintained in memory at any given time, thereby reducing the space complexity to $O(|B|)$.

[4]In practice, there are actually two lists – a forward and backward list – to accommodate for cases where the point is at the beginning or end of a subsequence.

significant reduction from the quadratic complexity required when evaluating the entire dynamic programming matrix.

4.4 Results and Discussion

The underlying hypothesis behind the behavioral distance metric proposed in the previous section is that the semantics of the network data and long-term sequences of activities provide a more robust notion of host behavior than measures that ignore such information. To evaluate this hypothesis, we compare our proposed dynamic time warp (DTW) metric to the well-known L_1 distance metric, which does not explicitly capture semantic and sequencing information. The experiments in our evaluation compare these two metrics using a broad range of analysis methodologies on a large dataset collected on a live network segment in an effort to pinpoint the benefits of our approach over naive metrics.

We begin by measuring the trade-off between speed and accuracy of our dynamic time warping-based distance calculation under varying settings of the multiplicative window parameter c. Using the results from this experiment, we choose a window that provides a balance between fidelity and speed, and show that our approach is feasible even on datasets containing millions of flows. Next, we compare our approach to a metric that ignores semantics and sequencing. In particular, we examine the clusters produced by a variety of cluster analysis techniques when using L_1 distance and our proposed DTW metric. Moreover, we look at the consistency of these behaviors across consecutive days of observation to show that our approach captures the most robust notion of network behaviors. Finally, we examine a concrete application of our metric to the problem of quantifying privacy of network hosts, and show how it enables application of well-known privacy definitions, such as the notion of (c, t)-isolation defined by Chawla *et al.*[13].

Data. In our evaluation, we make use of a dataset containing uni-directional flow records collected within the Computer Science and Engineering department of the University of Michigan. The CSE dataset, as we refer to it throughout this section, was collected over two consecutive twenty-four hour periods, and captures all local and outbound traffic from a single /24 subnet. The data contains 137 active hosts that represent a broad mix of network activities, including high-traffic web and mail servers, general purpose workstations, and hosts running specialized research projects. The relatively small number of hosts allows us to manually verify the behavioral information with system and network administrators at the University of Michigan. Table 4.1 provides a summary of the traffic within the CSE dataset on each of the observed days.

	Day 1	**Day 2**
Number of Hosts:	135	137
Total Flows:	14,400,974	16,249,269
Avg. Flows per Host:	106,674	118,608
Median Flows per Host:	4,670	2,955
Max Flows per Host:	6,357,810	6,620,785

Table 4.1: Traffic properties for both days of the University of Michigan CSE dataset.

Throughout all of our experiments, we consider six fields within the flow log data: start time, flow size, source IP and port, and destination IP and port. These fields are associated with the

Window Parameter c	**Avg. Time per Warp** in seconds (σ)	**Avg. Distance Increase** in percentage (σ)
25	1.0 (5.5)	5.0% (6.2%)
50	1.8 (10.1)	3.3% (4.5%)
100	3.2 (19.3)	1.9% (3.1%)
200	5.5 (36.2)	0.9% (1.9%)
500	11.1 (79.7)	0.0% (0.0%)

Table 4.2: Time vs. accuracy comparison, including averages and standard deviations for each window parameter tested. Accuracy measured as percentage increase in distances from window parameter $c = 500$.

appropriate distance metrics and used to measure behavioral distance according to the dynamic time warping procedure outlined in the previous section, except for instances where we explicitly make use of an alternate distance metric. Furthermore, we remove all non-TCP traffic from the dataset in an effort to minimize noise and backscatter caused by UDP and ICMP traffic. We note that since we are examining flow data, the use of TCP traffic should not unfairly bias our analysis since causal relationships among the flows will likely be dictated by application-layer protocols or other higher-order behaviors of the host.

Finally, we performed an initial analysis of the data using k-means clustering in combination with our DTW metric to pinpoint a variety of scanning activities, including pervasive Nessus scans [58] from two hosts within the CSE network and several IPs from China performing fingerprinting on CSE hosts. These scanning hosts were removed from the data to ensure that we focus our analysis on legitimate forms of network behaviors that may be verified via the University of Michigan CSE department's system and network administrators. The specifics of this technique are discussed in Section 4.4.2, but the fact that our method was able to pinpoint these scanning activities is interesting in and of itself.

4.4.1 Efficiency

Network datasets collected on real-world networks may contain millions of records. As such, it is important to understand the efficiency of our proposed metric and its ability to reasonably operate on large datasets. Moreover, we must also understand the impact of choosing the window parameter c on the accuracy of the resultant distance and the speed with which it is calculated. To do so, we choose five settings of the window parameter (25, 50, 100, 200, 500) and run distance measures among a random subset of 50% of the hosts from each day of the CSE dataset. The hosts in the sample selected for our experiment had an average of 63,984 flows each, with the largest host having 6,357,811 flows. For each of the window parameter settings, we measure the time it takes to perform the DTW procedure on a single 3.16GHz processor and the increase in distance from that which was calculated by the largest parameter setting (*i.e.,* $c = 500$), which are shown in Table 4.2.

The results of our efficiency experiments show that most parameter settings can perform DTW-based behavioral distance measures in a few seconds with relatively small changes to the overall distance, even when calculating distances among hosts with thousands or millions of flows. For example, with a parameter setting of $c = 100$, the DTW metric takes an average of only 3.2 seconds with an increase in the calculated distance of just under 2%. For the remainder of our evaluation, we fix the window parameter $c = 100$ since it appears to provide an appropriate trade-off

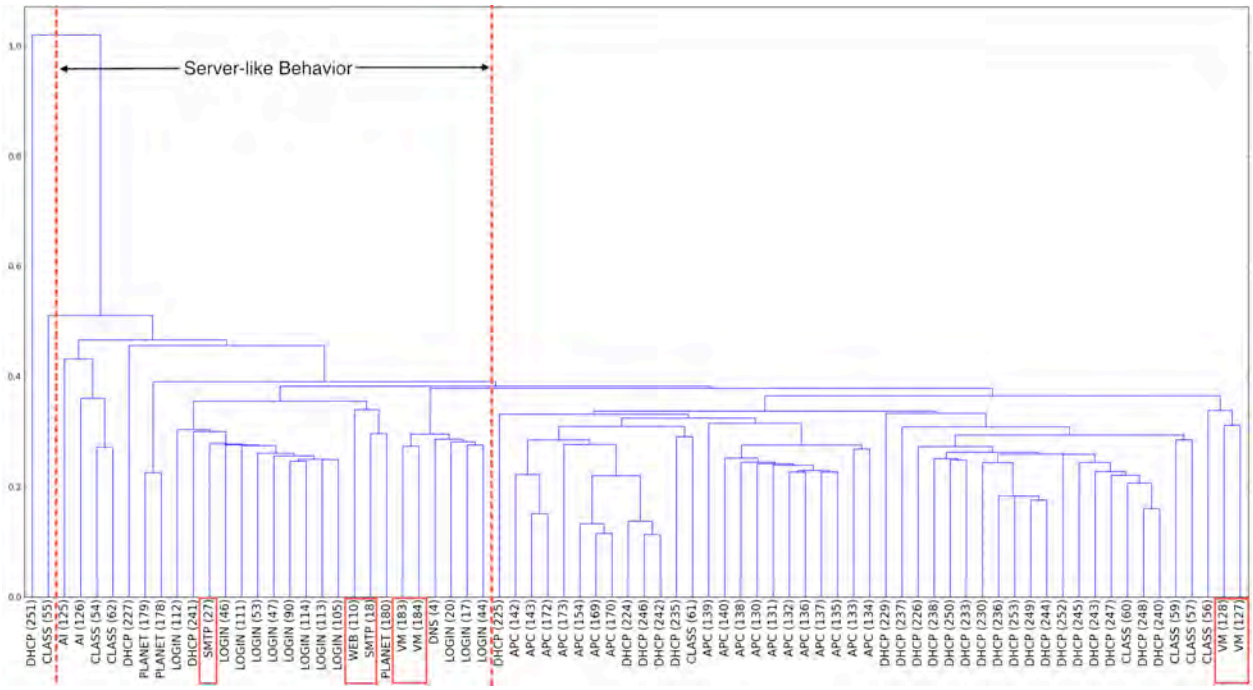


Figure 4.4: Dendrogram illustrating agglomerative clustering using DTW-based metric on hosts in the first day of the CSE dataset.

between distance calculation accuracy and speed.

4.4.2 Impact of Semantics and Causality

To evaluate the potential benefits of semantics and long-term causal information in behavioral metrics, we compare the performance of our DTW metric to the L_1 distance metric. In our experiments, the L_1 metric operates by creating distributions of values for each of the six fields, calculating the L_1 distance between the two hosts' respective distributions, and summing the distances. The DTW metric operates exactly as discussed in Section 4.3.1.

Our evaluation is broken into three parts. In the first, we use single-linkage agglomerative (*i.e.,* hierarchical) clustering to visualize and examine the behavioral similarity among all hosts in our experiments. The second experiment uses k-means clustering to explore the ability of the two metrics to produce clusters with coherent semantics. The final experiment compares the consistency of the clusters produced by the above techniques, as well as the similarity of the hosts across consecutive days to measure the robustness of the behavioral information captured by the two metrics. In the first two experiments, we examine only the first day of traffic from the CSE dataset, while both days are used in the final experiment.

In all of these experiments, we make use of information obtained from the University of Michigan CSE system and network administrators about the known usage of the hosts in our analyses to provide a general notion of the correctness of the clustering and to highlight specific cases for deeper inspection. This information allows us to label the hosts in our data with one of ten labels that describe the stated usage of the host when its IP was registered with the computer support staff.

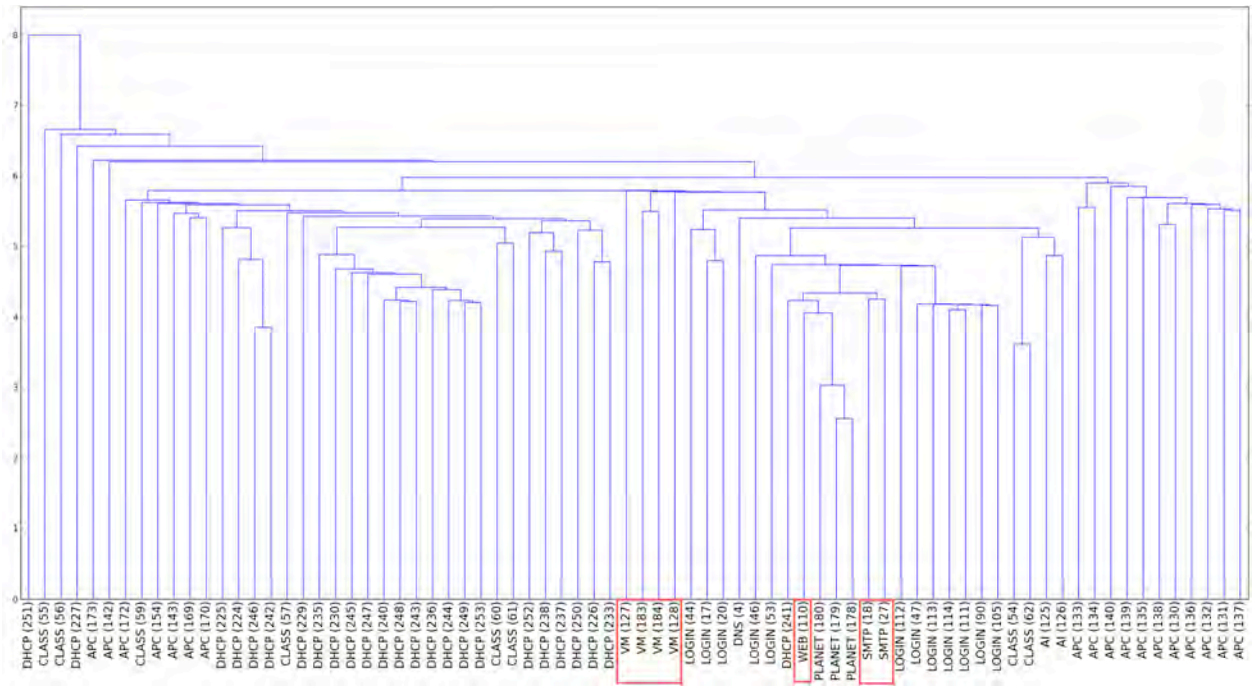


Figure 4.5: Dendrogram illustrating agglomerative clustering using L_1 distance on hosts in the first day of the CSE dataset.

These labels include: web server (WEB), mail server (SMTP), DNS server (DNS), a variety of host types involving general client activities (LOGIN, CLASS, DHCP), specialized research hosts for PlanetLab (PLANET), an artificial intelligence research project (AI), and auxiliary power units (APC). For the purposes of these clustering experiments, we only examine the subset of hosts that we have labels for (76 of the 137 hosts).

Agglomerative Clustering. The agglomerative clustering of hosts using the DTW and L_1 metric are shown as dendrograms in Figures 4.4 and 4.5, respectively. The dendrograms visualize the agglomerative clustering process, which begins with each host in its own cluster and then merges clusters iteratively with their nearest neighbor. The leaves in the dendrogram are labeled with the stated usage of the host obtained from system administrators and a unique identifier to facilitate comparison between dendrograms. The branches of the dendrogram illustrate the groupings of hosts, with shorter branches indicating higher levels of similarity.

At first glance, we see that both DTW and L_1 metrics group hosts with the same label in fairly close groups. In fact, it appears as though L_1 distance actually produces a better clustering according to these labels, for instance by grouping SMTP servers, PlanetLab hosts, and VM servers in very tight groupings. However, there are two subtle shortcomings that require deeper investigation. First, the L_1 dendrogram clearly indicates that there is relatively little separability among the various clusters, as evidenced by the small differences in branch lengths in the dendrogram from distances of 5 to 6.5. The impact of this is that even small changes in the underlying distributions will cause significant changes to the groupings. We will investigate this particular

		DTW Metric	L_1 Distance
Cluster 1	Hosts/Flows:	25 hosts/139.2 flows	12 hosts/165.8 flows
	Host Types:	APC 72%, DHCP 24%, CLASS 4%	APC 50%, DHCP 41%, CLASS 9%
	Activities:	$< SPORT = \{22, 6000\} >$ $< SPORT = 21, DADDR =$Chinese IP$>$ $< SPORT = 443, DADDR =$UMich DNS$>$	$< SPORT = \{22, 6000\} >$ $< SPORT = 21, DADDR =$Chinese IP$>$
Cluster 2	Hosts/Flows:	25 hosts/1,166.0 flows	18 hosts/908.2 flows
	Host Types:	DHCP 72%, CLASS 16%, VM 8%, AI 4%,	DHCP 89%, CLASS 11%
	Activities:	$< SPORT = \{22, 80\} >$	$< SPORT = 22 >$
Cluster 3	Hosts/Flows:	26 hosts/539,438 flows	46 hosts/305,211 flows
	Host Types:	LOGIN 46%, PLANET 11%, CLASS 11%, VM 8%, SMTP 8%, WEB 4%, AI 4%, DNS 4%, DHCP 4%	APC 26%, LOGIN 26%, CLASS 10%, DHCP 8%, VM 8%, PLANET 6%, AI 4%, SMTP 4%, WEB 2%, DNS 2%
	Activities:	$< SPORT = \{22, 25, 80, 111, 113, 993\} >$ $< SPORT = 5666, DADDR =$UMich DNS$>$	$< SPORT = \{21, 22, 80, 111\} >$ $< SPORT = 5666, DADDR =$UMich DNS$>$ $< SPORT = 443, DADDR =$UMich DNS$>$

Table 4.3: k-Means clustering of hosts in the first day of the CSE dataset using DTW and L_1 metrics. Activities represent the dominant state profiles obtained from applying our extension to the Xu *et al.*dominant state algorithm [88].

shortcoming during our consistency experiments. The second shortcoming is that while the labels provide a general idea of potential usage of the hosts, they say nothing about the actual activities being performed during recording of this dataset. As such, it is quite possible that hosts with the same labels can have wildly different behaviors.

To more closely examine the clustering provided by DTW and L_1 metrics, we manually observe two groups of hosts (highlighted in Figures 4.4 and 4.5): virtual machine hosts (VM) and major servers (SMTP and WEB). In the L_1 dendrogram, all VM hosts are grouped together, whereas in the DTW dendrogram the hosts are grouped into pairs. Moreover, these pairs are on opposite sides of the dendrogram indicating significant differences in behavior. When we manually examine these hosts' activities, we see that one pair (127,128) appear to be performing typical client activities, such as outgoing SSH and web connections, with only approximately 2,000 flows each. The other pair (183,184), however, had absolutely no client activities, and instead most of the approximately 6,000 flows consisted of VMWare management traffic or Nagios system monitoring traffic [55]. Clearly, these are very distinct behaviors – client activity and basic system management activity – and yet the L_1 metric was unable to distinguish them.

For the primary servers in the dataset, the L_1 distance metric groups the two SMTP servers together, however our DTW metric ends up grouping one of the SMTP servers (18) with the web server while the second SMTP server is much further away. Upon closer examination, the SMTP servers are differentiated by the fact that one server (18) is the primary mail server that receives about five times the amount of the traffic as the second server (27). In addition, the first server (18) receives the majority of its connections from IPs external to the CSE network, while the second server (27) receives many of its mail connections from hosts within the CSE IP prefix and has a significant number of connections from hosts performing SSH password dictionary attacks. It is this local vs. external preference that causes the first server (18) to be grouped with the web server,

since the web server also has a significant amount of traffic, almost all of which is associated with external hosts. Naively, the L_1 clustering appears to make the most sense given these labels, but again its lack of semantic and causal information has caused it to ignore the actual behaviors.

As a side effect of our close examination of some hosts within the dendrogram, we also gained some insight into the general structure of the dendrogram in the case of the DTW metric. That is, the DTW metric produced a dendrogram where the hosts on the right side of the dendrogram perform general client activities, while most hosts on the left side act as servers. This separation is illustrated by vertical dotted lines in Figure 4.4. The DTW dendrogram is further refined by the type of hosts (internal vs. external) that they communicate with, the volume of traffic, and other interesting behavioral properties.

k-Means Clustering. Another way to evaluate the performance of the DTW and L_1 metric is to consider if the clusters produced maintain some level of behavioral coherence, and whether the groupings are similar to those that might be produced by a human network analyst looking at the same data. To examine these properties, we use k-means++ clustering [2] to produce clusters from each of the distance metrics and then examine the properties of the resultant clusters. In particular, we use the dominant state analysis technique of Xu *et al.*[88] to characterize the dominant behaviors in each cluster, examine the distribution of host labels in each cluster, and manually examine a random sample of each cluster to determine the overall behavior being captured.

While a full discussion of the Xu *et al.*dominant state algorithm is beyond the scope of this evaluation, we provide a high-level notion of its operation and how we extend it to capture behaviors of groups rather than individual hosts. The algorithm begins by calculating the normalized entropy of each field being analyzed, and then ordering those fields from smallest entropy value to greatest. The algorithm then selects all values from the smallest entropy field whose probability are greater than a predetermined threshold t, thereby creating "profiles" for each value. Then, each of those profiles is extended by examining values in the next smallest entropy field whose joint probability of occurrence with the profile values is above the threshold t. The profiles are extended iteratively until no new values may be added from the current field, at which point they are considered to be final profiles.

In the original paper, Xu *et al.*defined the distribution of values on a per-host basis to quickly determine host activities. To apply the same procedure to groups of hosts, we apply the dominant state algorithm in two levels. At the first level, we run the algorithm on distributions of values for each host in the cluster. This produces dominant state profiles for each of those hosts. From these profiles, we extract the values for each of the present fields to create new distributions. These distribution represent the probability of a value occurring in the dominant state profiles for the hosts in the current cluster. Finally, we apply the dominant state algorithm to these cluster-wide distributions to extract out the profiles that occur most consistently among all hosts in the clusters.

For this experiment, we manually examined a sample of hosts throughout the entire dataset and determined that there were, roughly, three high-level classes of activities: server activities, client activities, and noise/scanning activities. This manual classification was supported by the results of our agglomerative clustering analysis, which showed several levels of increasingly subtle behavioral differences within each of these three classes. Given this rough classification of behavior, we set $k = 3$ and see if the resultant clustering indeed captured the intuitive understanding of the activities as determined by a human analyst. Table 4.3 shows the break down of the three clusters

Host Type	Num. Hosts	DTW Metric		L_1 Distance	
		Num. Perfect Consistency	Avg. Rank	Num. Perfect Consistency	Avg. Rank
WEB	1	1	1.0	1	1.0
DNS	1	1	1.0	0	17.0
PLANET	3	2	1.3	3	1.0
VM	4	2	1.5	0	41.3
LOGIN	12	8	2.1	8	14.2
SMTP	2	1	3.5	2	1.0
APC	18	2	16.2	0	39.9
AI	2	0	41.5	0	40.5
DHCP	24	2	48.4	1	31.8
CLASS	8	1	54.4	0	31.1
TOTAL	75	20	17.1	15	21.9

Table 4.4: Host behavioral consistency for hosts occurring in both days of the CSE dataset.

produced by the k-means++ algorithm using DTW and L_1 metrics. The results of the DTW metric clustering shows that the clusters are indeed broken into the three classes of activity found via manual inspection. Cluster 1, which contains primarily the power supply devices (APC) and DHCP hosts, had significant scanning activity from IP addresses in China, and relatively low traffic volumes indicating only sporadic use. By comparison, hosts in Cluster 2 exhibit traditional client behaviors of significant SSH and web browsing activities. Finally, Cluster 3 contains hosts with significant server-like activities. For instance, the hosts related to the artificial intelligence research project (AI) were found to be running web servers that provide statistics to participants in the project, and most of its activity is made up of web requests.

The L_1 metric, on the other hand, produced somewhat incoherent clusters. While there is some overlap in the clusters and observed behaviors, it is clear by the distribution of host types that these clusters mix behaviors. The most prominent example of this is that many of the power supply hosts (APC) that we verified as have low traffic volumes and scanning activity were grouped in with the server cluster (Cluster 3). Moreover, the client cluster (Cluster 2) contains many of the DHCP hosts with scanning activity, which in turn alters the behavioral profile for that cluster to remove web activities. The results of this experiment certainly indicate that the L_1 metric simply does not capture a coherent notion of behavior. Moreover, when the number of clusters is increased, the differences between the DTW and L_1 metrics become more significant. That is, the L_1 metric continues to mix behaviors, while the DTW metric creates clusters that represent increasingly fine-grained behaviors, such as the preference for communicating with internal versus external hosts.

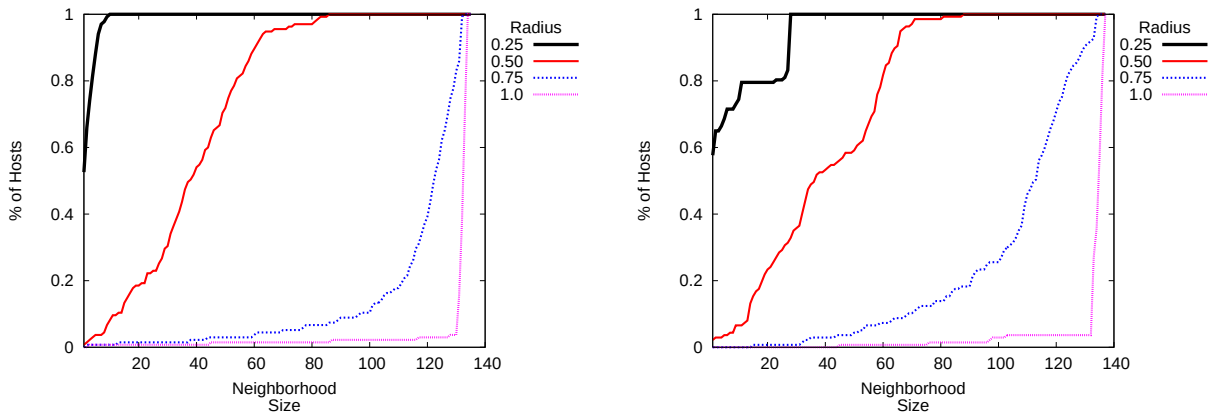
As mentioned earlier in this section, this clustering approach was also used to filter a large portion of the scan traffic found in the dataset. Specifically, when we first ran this experiment, we found that the behavioral profiles produced by Cluster 1 were consistent with a wide range of scan traffic, and that the number of hosts in that cluster were significantly larger than expected. We were then able to use the behavioral profiles to remove traffic from scanning IPs and produce a much cleaner clustering without most of the initial scanning activity. As you can see by the scanning IP from China found in the profile of Cluster 1, it appears that we can continue performing this clustering approach to iteratively identify increasingly subtle scanning activity.

Changes in Behavior Over Time. Perhaps one of the most important properties of any behavioral metric is its robustness to small changes in underlying network activity. That is, we want any changes in the measured distance to be related to some high-level behavioral change and not minute changes in the specifics of the traffic. As our final experiment, we use both days of traffic in the CSE dataset to determine the sensitivity of the DTW and L_1 metrics to changes in behavior over time. To do so, we take the set of all hosts that occur in both days (as determined by IP address), and compare their day one time series to those of all hosts in day two. This produces a list of distances for each host in the day one data to the day two hosts. With consistent behavior and a a behavioral metric that is robust to minute changes in activity, we would expect to find that each day one host is closest to itself in the day two data. Of course, there are also instances where host behavior did significantly change between the two observation periods. Therefore, our analysis looks at both the number of hosts within each label class whose behaviors appear to remain the same and the reasons that some hosts' behaviors apparently change.

To begin, we provide a summary of the above consistency experiment for each host label when using the DTW and L_1 metrics in Table 4.4. The table shows two values: the number of hosts with perfect consistency and the average consistency rank of all hosts in the group. By perfect consistency, we mean hosts in day one whose closest host in day two is itself. Consistency rank refers to the rank of the day one host in the sorted list of day two distances. Ideally, if the behaviors of the hosts in the group are exactly the same we would have all hosts with perfect consistency and an average rank of 1.0. The most obvious observation we can make from the results of this experiment is that the L_1 metric appears to be more brittle than DTW with a significantly greater average rank and fewer perfect consistency hosts overall. What is more interesting, however, are the cases where DTW indicates change in behavior and L_1 does not, and vice versa.

For the case where DTW indicates change and L_1 does not, we again examine the two SMTP servers found in our data. Recall from the previous clustering experiments that one SMTP server in the first day of data acts as the primary server with many connections to external hosts, while the other receives many fewer connections and those are primarily to internal hosts. Upon closer inspection of the two servers' behaviors in day two, we find that the primary SMTP server's (18) activities are effectively unchanged. The secondary server (27), however, has a significant increase in the proportion of traffic that is related to mail activities. In particular, the secondary SMTP server (27) in the first day of data has a roughly even split between general mail traffic and an SSH password dictionary attack (*i.e.,* hundreds of consecutive SSH login attempts), while in the second day the SSH attacks stop almost completely and the general mail traffic to both internal and external hosts increase significantly. Intuitively, this does indeed indicate a change in behavior due to the presence of the SSH attack and change in mail activity, although the L_1 metric indicates that no such change took place.

Next, we look at the group of VM servers for the case where the L_1 metric indicates a change and the DTW metric does not. In this case, manual inspection of all four VM hosts indicates that there were no significant changes in traffic volume or activities for any host. The only noticeable change was that one of the client-like VMs (127) was port scanned for a few seconds late at night on the second day. Given this information, our DTW metric's ranking makes perfect sense – the two VMs running management protocols (183,184) were exactly the same as their previous day's activity, while the client-like VMs (127,128) got confused for one another in the previous day which is evidenced by the average ranking of 1.5 for that group (*i.e.,* rank of 1.0 for the management



(a) Day 1

(b) Day 2

Figure 4.6: Privacy neighborhood sizes for different distance radius settings.

VMs, 2.0 for the client VMs). With the L_1 metric, not only did it confuse the VM behaviors for one another in the previous day, but it also confused it with dozens of other client-like hosts (*i.e.,* DHCP, LOGIN, etc.) and management hosts (*i.e.,* APC), thereby causing the significantly higher average rank for that group. In fact, the L_1 metric only seems to achieve a lower average rank in groups where changes in behavior are expected, such as the DHCP, APC, or CLASS groups, and where the the rankings are likely to improve by chance due simply to tiny changes in activities.

4.4.3 Applications

To conclude our evaluation of the proposed DTW behavioral metric, we discuss its potential applications. Certainly, the fact that our proposed metric provides a robust, semantically meaningful notion of host behavioral similarity means that it may be a good foundation for a number of network analysis tasks, such as anomaly detection or host classification. In fact, the cluster analysis technique described above can be thought of as a form of host classification and an anomaly identification method. There are, however, some non-obvious forms of network data analysis which may benefit from our more formalized approach. One such application is the use of the DTW metric as the basis for analyzing the privacy of hosts within anonymized network data.

Roughly speaking, anonymized network data is simply a transformation of data collected on a computer network such that certain sensitive information is not revealed to those who use the data, but the data remains generally useful to researchers and network analysts. This sensitive information includes learning the real identities of hosts found within the data despite those identities being replaced with a pseudonym (*e.g.,* prefix-preserving anonymization [34, 63]). One of the most difficult parts of achieving network data anonymization is the lack of applicable privacy definitions upon which these transformations can be based, due primarily to the fact that rigorously defining the behavior of hosts and the ways these behaviors may change remains an open problem. Although prior works (including our own) have attempted to define privacy analysis techniques for anonymized data [24, 70], their methods have been based upon techniques like the L_1 distance metric evaluated above, and consequently are likely to provide a rather loose privacy analysis. We argue that given the DTW metric's unique ability to capture a robust notion of host behavior

and the fact that it embeds this behavior in a well-defined metric space, it holds good promise in bringing formal privacy definitions to the field of anonymized network data.

We illustrate this point by examining one way in which a privacy analysis for network data may be built around the definition of (c, t)-isolation put forth by Chawla *et al.*in the context of multi-dimensional, real-valued spaces [13]. This privacy notion essentially states that any "anonymized" point should have t real points from the original data within a ball of radius proportional to c centered at the anonymized point. That is, each anonymized point should have a conserved neighborhood of real points that it may "blend in" with, and therefore provide the potential adversary with some level of uncertainty about the point's real identity. The definition can be used as the basis of an analysis methodology simply by looking at the distribution of neighborhood sizes for each anonymized point at increasing radius sizes – such information may be used by a data publisher to, for instance, determine the relative risk of publishing the anonymized data in its current form. The distribution may be visualized via a cumulative distribution function that shows the percentage of points with a number of neighbors (*i.e.,* neighborhood size) less than the given value for a specified radius.

To adapt the definition and analysis methodology for network hosts, we simply consider the entire time series for the hosts to be a "point" and use the DTW metric to calculate the radius. Figure 4.6 shows examples of this privacy analysis methodology applied to hosts within the the two days of our CSE dataset under the assumption that none of the six fields in our metric have been altered by the anonymization process. One way of interpreting this analysis is that the radius bounds the potential error in the adversary's knowledge of the host's behaviors, while the number of hosts within the neighborhood provides a sense of the privacy of that host. Therefore, if a data publisher assumes the adversary could gain significant knowledge of a host's behaviors, perhaps derived from publicly available information, it would be prudent to consider the neighborhood sizes when the radius is relatively small. As Figure 4.6(a) shows, for example, even for a radius of 0.5, well over 80% of the hosts in the data have neighborhoods of size 20 or greater, thereby indicating potentially significant privacy for those hosts. Of course, as discussed earlier in Section 4.3.2, this is contingent upon the assumptions made about the adversary's "view" of the data via the metric definitions.

An obvious downside of this approach is that it is difficult to interpret the semantics of the overall DTW distances. That is, understanding what exactly a radius of 0.5 means with respect to the underlying behaviors may be difficult. One way to address this issue is to provide distributions of distances for each dimension computed during the time warping process, in addition to the Euclidean distance. Additionally, the semantically-meaningful metric spaces for each field may also need to be altered to accommodate for measuring behavioral distance between fields that have been altered by the anonymization process in the anonymized network data and those in the original (*e.g.,* comparing prefix-preserving IP pseudonyms to the original IPs). However, we believe that the fact that our approach allows for such changes to the underlying semantics is a contribution in and of itself.

4.5 Recommendations

As a whole, these results indicate that it is useful to consider long-term temporal characteristics of network hosts, as well as the semantics of the underlying network data when measuring behavioral

similarity. Furthermore, our results point toward several potentially interesting areas of future work. In the short term, one may consider the development of more refined distance metrics, including fine-grained metric spaces for a wider range of data types and time warping methods that allow for localized reordering of points. The results also call for a study of the performance of our DTW metric when applied to non-TCP protocols and to network objects other than hosts, such as web pages. More generally, a more formal method for characterizing behaviors, which may be used as the basis for provable network data anonymization techniques or robust traffic generation methods, seems warranted.

4.6 Conclusion

Many types of network data analysis rely on well-known distance metrics to adequately capture a meaningful notion of the behavioral similarity among network hosts. Despite the importance of these metrics in ensuring sound analysis practices, there has been relatively little research on the impact of using generic distance metrics and ignoring long-term temporal characteristics on analysis tasks. Rather, distance metrics used in practice tend to take a simplistic view of network data by assuming they inherit the semantics of its syntactic representation (*e.g.,* 16- or 32-bit integers), or that those values have no relationship at all. Moreover, they examine network activities in isolation or within short windows (*e.g., n*-gram distributions), which removes much of the long-term causal information found in the data. Consequently, these approaches are likely to provide brittle or unrealistic metrics for host behavior.

In this report, we explored an alternative approach to defining host similarity that attempts to incorporate semantically meaningful spatial analysis of network activities and long-term temporal sequencing information into a single, unified metric space that describes host behaviors. To accomplish this goal, we developed metric spaces for several prevalent network data types, showed how to combine the metric spaces to measure the spatial characteristics of individual network data records, and finally proposed a method of measuring host behavior using dynamic time warping (DTW) methods. At each stage in the development of this framework, we brought to light potential pitfalls and attempted to explain the unique requirements surrounding the analysis of network data, including the need to carefully define normalization procedures and consider assumptions about the data made in developing the metrics. Our proposed metric was evaluated against the well-known L_1 distance metric, which ignores both semantic and temporal characteristics of the data, by applying cluster analysis techniques to a dataset containing a variety of realistic network host activities. Despite the admitted simplicity of our example metrics, the results of these experiments showed that our approach provides more consistent and useful characterizations of host behavior than the L_1 metric.

Chapter 5

Amplifying Limited Expert Input to Sanitize Large Network Traces

5.1 Introduction

Visibility into packet payloads supports numerous network security defenses and dependability analyses. For example, technologies ranging from simple signature-based intrusion detection systems (e.g., Snort, `www.snort.org`) to advanced techniques for developing exploit signatures require access to network packet payloads (e.g., [85]). To evaluate the efficacy of such proposed defenses, therefore, it is necessary to have access to payload-bearing network traffic traces on which to test them.

While there has been significant progress in the release of other types of network traffic traces for research purposes in the last decade, the release of packet payloads remains severely limited due to privacy concerns, and this continues to hamper research progress in numerous types of network defense and performance tests. Packet payloads can contain sensitive information ranging from personal user information to security-relevant data about network topology and service configuration. Thus, data publishers that release packet traces must first sanitize the traces by removing the sensitive information — and in virtually every case, this sanitization includes deleting the payloads in their entirety. This obviously destroys the utility of the trace for research that requires packet payloads.

The extreme rarity with which payload data is released is due to numerous challenges that data publishers face in trying to sanitize packet payloads. First, almost any interesting trace contains too many packets for an administrator to examine exhaustively. Second, even if the trace contains packets of only one protocol (e.g., selected by filtering on ports), the packet formats within that protocol may be numerous or, even worse, undocumented. For example, the Samba project required many researchers' efforts over several years to reverse engineer the file-sharing protocol in Microsoft Windows networks, for which the protocol specification was not released to the public. Third, packet payloads may contain many types of information that may be deemed sensitive, e.g., user names, IP addresses, passwords, host names, and a range of user-generated content. Indeed, the sheer diversity of content that one might deem as sensitive in a free-form protocol like HTTP is overwhelming.

As a step forward in this space, in this paper we propose a framework and tool to support

packet trace sanitization. To accommodate incompletely documented protocols, our framework is built around a human *expert* who can explore selected protocol packets well enough to accurately identify the sensitive information they contain. However, since dataset release is rarely a business priority, we presume that this expert has very little time to devote to this effort. For this reason, we structure our framework hierarchically, using the expert's input to select others from a set of candidate *workers*, based on their abilities to mark sensitive data in packets similarly to the expert. These selected workers then mark sensitive data in a small group of packets that can best represent the characteristics of the overall dataset, which our technique then applies to automatically identify sensitive data of the remaining packets in that dataset. We stress that the expert is involved only in marking sensitive data for a very small number of packets — generally far fewer than the total number of packet formats available in the protocol. As such, we cannot impose upon the expert to analyze even one packet of each format, and of course, we may not even know how many formats there are due to the unavailability of the protocol specification.

At the core of this technique is an algorithm that selects and presents packet data to the workers in a fashion that best enables them to identify fields that they deem sensitive and then to extrapolate from those inputs on that selected data to sanitize the entire dataset. (The expert examines only a small subset of the representative packets selected for the workers.) Doing so in a way that achieves good accuracy in sanitizing the whole dataset requires that our technique (i) judiciously select the data that the workers will examine; (ii) organize the presentation of the selected data to maximize each one's competence in identifying sensitive fields; and (iii) effectively draw inferences from their inputs to sanitize fields in packet data not presented to them directly.

At a high level, our technique accomplishes these goals through a multistep process. Packets in the trace are first divided into contiguous tokens, each with a type. Stratified sampling is applied to these typed token sequences in order to select a fraction of the packets for further analysis, while minimizing the likelihood of excluding any particular packet type. These selected packets are clustered into groups with similar structure; intuitively (and ideally), these clusters correspond to packet formats. We then select representatives from each cluster, which we present to a worker in an aligned fashion so as to best reveal their common structures, a technique known to accelerate visual recognition of homogeneous structures [30]. The best workers are selected from a group of workers by comparing each one's performance to that of the expert in marking a small subset of representatives; the selected workers then mark sensitive fields in all representatives. After the workers identify sensitive tokens in the representatives for each cluster, these identified tokens are mapped onto the entire dataset, in order to identify sensitive tokens in packets that the workers never examined.

We describe our design and implementation of a tool that implements this approach and present an evaluation based on a user study involving professional network administrators as workers. Our results show that both clustering and alignment have a statistically significant, positive effect on their abilities to identify sensitive data and that the effects of the two are additive. The study also reveals that even network administrators show substantial variability in their abilities to locate sensitive information in network data, underscoring the difficulty of the task at hand and the need to down-select workers on the basis of the similarity of their markings to an expert's. We show that combining the sensitivity determinations of the two best workers and using these to mark the entire dataset identifies sensitive data in the original dataset very well. Lastly, we use an entropy-based approach to quantitatively evaluate the ability of an adversary to infer the contents of fields

sanitized using our technique.

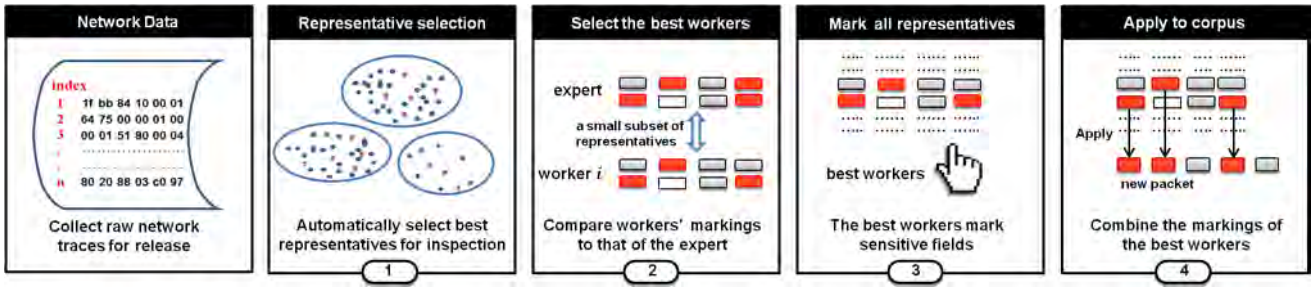


Figure 5.1: Framework of network trace sanitization by leveraging best worker input

5.2 Methods, Assumptions, and Procedures

At the core of our approach to identifying sensitive information in packet payloads is an algorithm that selects a subset of packet payloads to present to workers (a subset of which is also presented to the expert). At a high level, this algorithm can be viewed as a natural application of clustering and sequence alignment techniques for assisting workers in more readily identifying sensitive information in network data. Our proposed solution requires that we first tokenize the payloads of the packets by labeling them in a more compact representation composed of generic types of fields. Next, we cluster the tokenized packets in order to organize them roughly according to their formats.

Obviously, presenting a large corpus of data—even in an aligned form—to a human being, and expecting her to effectively sift through such data would not be a fruitful task, to say the least. In order to ease the arduous job of finding potentially sensitive information in a large corpus of data, we present to the workers only representatives for each cluster that capture the most mutual information (and that have low redundancy). As the worker marks tokens in the view displayed to her (e.g., by highlighting regions within the visually-aligned representatives), these annotations are recorded. Once the interactive session has completed, her selections made during the process are then used to automatically infer other sensitive tokens in the remainder of the corpus, without further participation from the worker. The overall process is depicted in Figure 5.1. We discuss the specifics of how we select representatives from the corpus in more detail in Sections 5.2.1–5.2.5 and then discuss how we use the tokens marked sensitive by a worker to identify sensitive fields in the larger corpus in Section 5.2.6.

5.2.1 Tokenization

We remind the reader that in order to be protocol agnostic, we deliberately assume no prior knowledge of the protocol format, and so before we can select the best candidates to present to workers, we must first tokenize the data. To do so, we abstract each packet by grouping its bytes into tokens, each of a certain type. The token types we use take advantage of a growing body of work on protocol reverse engineering (e.g., [27, 46]) that suggests suitable token types for text protocols (e.g., HTTP, FTP), binary protocols(e.g., DNS, DHCP) and so-called hybrid protocols (e.g., SMB). Specifically, our three token types are: (1) **Length** fields: consecutive

printable characters proceeded by a byte value indicating the number of characters to follow. Both the printable characters and the byte are combined into a single length field. (2) **Text** fields: several consecutive printable characters, the length of which is greater than some threshold.[1] And (3) **Binary** fields: any single byte except those defined as a length field or text field. Each packet is tokenized by scanning the payload from beginning to end. Figure 5.2 shows an example of the tokenized representation of two packets. For the remainder of the paper, all subsequent operations are on tokenized sequences, and we denote the tokenized sequence of a packet pkt by tokenize(pkt).

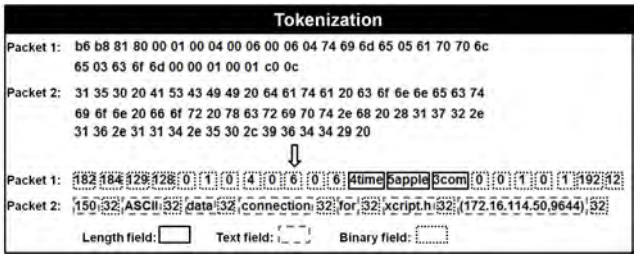


Figure 5.2: Example tokenization of two packet payloads.

5.2.2 Sampling the Data

For improved performance, we next sample packets from the entire dataset and use only the selected packets in subsequent stages. However, sampling packets uniformly at random risks omitting packet formats present in the full dataset, especially if those formats are rare. To overcome this obstacle, we use *stratified sampling* [18, 59] in order to preserve the diversity of the entire dataset. Specifically, we partition the tokenized sequences into homogeneous subgroups; lacking information about packet semantics or formats, we simply partition sequences according to their lengths (i.e., the number of tokens in each). We then draw a random sample from each stratum of size proportional to the stratum size, i.e., the number of sequences it contains.

5.2.3 Grouping Similar Packet Formats

Given a selection of sequences (i.e., tokenized packets), the next task is to effectively cluster these sequences into groups representing different packet formats. To do so, we first define a distance between sequences, and then provide an algorithm for performing clustering using those distances.

Our chosen distance is based on sequence alignment [37]. That is, to define a distance between two packets, we first find the optimal *token-based sequence alignment* of the packets, which is an alignment of the pair of sequences of tokens corresponding to those packets. Specifically, for an alignment of two token sequences, each aligned pair of tokens is assigned a positive score (an "award") if they match, and a negative score (a "penalty") if they are a mismatch or if one of them is a gap inserted by the alignment process. We use $\mathsf{Penalty}_{\mathsf{mis}}$ and $\mathsf{Penalty}_{\mathsf{gap}}$ to represent the mismatch and gap penalties, respectively. In assigning awards for any two matching tokens, we not only consider their types, but also consider their values. In particular, if two tokens have the same type and value, we assign a larger matching score $\mathsf{Award}_{\mathsf{val}}$. If they are of the same type but

[1] Similar to [27], we choose a threshold of 3 printable characters.

different values, we assign a smaller matching score $\mathsf{Award}_{\mathsf{typ}}$. The overall alignment score between packets pkt, pkt', denoted
$\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}, \mathsf{pkt}')$, is computed using the matching scores, mismatch penalties and gap penalties. Our scoring function is formulated as $\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}, \mathsf{pkt}') = \mathsf{N}_{\mathsf{val}} \times \mathsf{Award}_{\mathsf{val}} + \mathsf{N}_{\mathsf{typ}} \times \mathsf{Award}_{\mathsf{typ}} + \mathsf{N}_{\mathsf{mis}} \times \mathsf{Penalty}_{\mathsf{mis}} + \mathsf{N}_{\mathsf{gap}} \times \mathsf{Penalty}_{\mathsf{gap}}$, where $\mathsf{N}_{\mathsf{val}}$, $\mathsf{N}_{\mathsf{typ}}$, $\mathsf{N}_{\mathsf{mis}}$ and $\mathsf{N}_{\mathsf{gap}}$ correspond to the number of tokens with matched values, tokens with matched types, mismatched tokens and inserted gaps, respectively, for an alignment of $\mathsf{tokenize}(\mathsf{pkt})$ and $\mathsf{tokenize}(\mathsf{pkt}')$ that maximizes $\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}, \mathsf{pkt}')$. We then define the distance as

$$\mathsf{dist}(\mathsf{pkt}, \mathsf{pkt}') = 1 - \frac{\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}, \mathsf{pkt}')}{\mathsf{Score}_{\mathsf{max}}(\mathsf{pkt}, \mathsf{pkt}')} \tag{5.1}$$

where $\mathsf{Score}_{\mathsf{max}}(\mathsf{pkt}, \mathsf{pkt}') = \max\{\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}, \mathsf{pkt}),$
$\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}', \mathsf{pkt}')\}$ denotes that largest possible value of $\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}, \mathsf{pkt}')$. The optimal sequence alignment for two token sequences, and hence the distance (5.1), can be computed efficiently using the well-known Needleman-Wunsch algorithm [57].

Clustering method Given this distance calculation, we apply iterative K-medoids clustering [39] to partition the packet sequences into different clusters. Unlike K-means, which takes the arithmetic mean of each cluster's points as its centroid, the K-medoids algorithm chooses a member of the cluster as its centroid, namely that which minimizes the average distance to all cluster members. Instead of deciding the number of clusters in advance, we employ the following algorithm to iteratively grow the number of clusters, which is parametrized by a value r, $0 < r < 1$:

1. Assign all packets into one cluster, and find the medoid of the cluster.
2. For each cluster, find the packet furthest from its medoid as a candidate for a new medoid. Choose the furthest one among these candidates as the medoid of a new cluster.
3. Re-cluster all packets, assigning each packet to the closest existing medoid. After that, recompute the medoid of each cluster.
4. Repeat steps 2 and 3 until no packet is further than r times the average medoid-to-medoid distance.

Therefore, the number of generated clusters depends on the input parameter r.

5.2.4 Alignment of Packets

The clusters output from the procedure described in Section 5.2.3 are the clusters from which our algorithm selects representatives (with at least one representative being selected from each cluster). In preparation for performing this selection, we first align all of the packets in each cluster collectively. We now detail how the alignment is done, and discuss the selection of representatives later.

A key challenge in performing multiple sequence alignment in our setting is the fact that we may need to operate over clusters with thousands of packets. For efficiency, we use a progressive method, which generates an alignment by first aligning the most similar sequences and then successively

adding less similar sequences to the growing alignment until all packets in the cluster have been incorporated.

With progressive alignment, the quality of the final alignment generally depends on the order with which the sequences are incorporated. To determine this order, we treat the cluster as a graph with vertices being the packets and an edge between each pair of packets weighted by their distance (5.1). We then use Prim's algorithm [67] to create a minimum spanning tree, and integrate (i.e., align) the vertices together in the order in which they are included in the tree. Since Prim's algorithm adds vertices in increasing order of their distance to their nearest vertex already in the tree, aligning vertices in this order should intuitively delay the insertion of gaps in the overall alignment as long as possible (and hopefully render most gaps unnecessary).

The gaps inserted by align, in locations indicated by
isNewGap[ ], are then propagated to the aligned forms of the packets already integrated into the tree, i.e., by inserting the gaps into the same positions in those sequences.

The behavior of align differs in an important way from sequence alignment as described in Section 5.2.3. To avoid the introduction of gaps into the aligned sequences seq$[k][\,]$ to the extent possible, we alter align to (i) output in consensusSeq$[j]$ the *disjunction* of the tokens from inputs consensusSeq[ ] and tokenize(pkt*) that it aligns to position j (under an optimal alignment as defined in Section 5.2.3); and (ii) assign a matching award (in value or type) to tokens if the token of tokenize(pkt*) matches *any disjunct* of the token of input consensusSeq[ ] (and where a gap in consensusSeq[ ] type-matches nothing). Consequently, after propagating gaps to the other sequences already integrated into the tree, an invariant of the loop is that consensusSeq$[j] = \bigvee_{k=1}^{\mathsf{mstSize}}$ seq$[k][j]$. Figure 5.3 shows the result of aligning a cluster of ten packets using this algorithm. Notice that fields with similar type or value have been correctly aligned.

5.2.5 Selecting Representatives for Inspection

At this stage in our overall process, the elements of each cluster are aligned in equal-length sequences denoted as
seq$[1 \ldots |\mathsf{cluster}|][\,]$. To select representatives and present them to the worker for inspection, we borrow a technique from Pan et al. [62]. The input required by this technique is a collection of equal-length feature vectors. To generate this input, we simply define feature vectors seqFV$[1 \ldots |\mathsf{cluster}|][\,]$ so that seqFV$[k][i] = 0$ if seq$[k][i] =$ gap and seqFV$[k][i] = 1$ otherwise. Due to the alignment of seq$[1 \ldots |\mathsf{cluster}|][\,]$, the tokens of all sequences at position i typically have the same type, and so a binary gap/no gap representation for these feature vectors suffices. The technique of Pan et al. then uses these feature vectors to select representatives that maximize their mutual information and minimize their redundancy.

Figure 5.3 shows representative sequences generated for a particular cluster. The most closely similar sequences, grouped via the method of Pan et al. [62], are labeled with the same shape (e.g., star). One representative is picked per group of similar sequences, based on maximizing the mutual information and minimizing redundancy. In this way, the approach for selecting representatives improves the efficiency of worker inspection by dramatically decreasing the number of sequences presented for inspection.

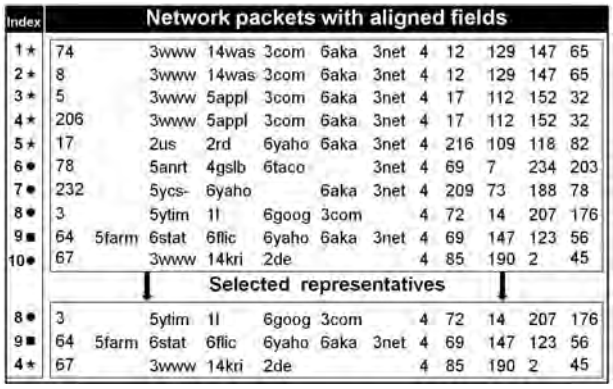


Figure 5.3: Example of representative selection; each representative and its most similar packets are denoted by the same shape (e.g., star).

5.2.6 Applying Worker Feedback

The representatives selected as described in Section 5.2.5 are presented to a group of workers, via an interface such as that described in the appendix. Our technique does not require a specific interface, though it should present the representatives to the worker in a way that promotes the identification of sensitive fields and that provides the worker an ability to mark which fields she believes to be sensitive. In Section 5.3.2, we evaluate two features of such a user interface that we believe, based on previous findings about user perception [3, 30], can ease the worker's task, namely presenting similar representatives from one cluster at a time and presenting tokenized representatives in their aligned form.

The distinct capacities of the workers in identifying sensitive fields make it challenging to apply their markings to the full dataset — recruited workers may have diverse skill levels and training. This motivates our attempt to achieve better accuracy by leveraging inputs from only the most skilled in the worker pool. To identify these most skilled workers, we compare the fields indicated as sensitive by each worker in a small subset of the representatives that worker examined, to ground truth for those representatives as determined by the expert. We make this comparison on the basis of standard measures, namely precision and recall:

$$\mathsf{Precision} = \frac{|\{\mathsf{identified\ fields}\} \bigcap \{\mathsf{sensitive\ fields}\}|}{|\{\mathsf{identified\ fields}\}|}$$

$$\mathsf{Recall} = \frac{|\{\mathsf{identified\ fields}\} \bigcap \{\mathsf{sensitive\ fields}\}|}{|\{\mathsf{sensitive\ fields}\}|}$$

To select the *best* workers, however, it is necessary to reduce these two measures to one, on which worker performance can be ordered. For this, we use the F-score [72] statistic, which computes a weighted average of recall and precision:

$$\mathsf{F}_\alpha = (1+\alpha^2) \cdot \frac{\mathsf{Precision} \cdot \mathsf{Recall}}{\alpha^2 \cdot \mathsf{Precision} + \mathsf{Recall}}$$

Essentially, F_α measures the effectiveness of identification with respect to the participant who places

α times as much importance to recall as precision [72]. (We will comment on the values of α we employ in Section 5.3.)

Those workers with the highest F-scores on these representatives are selected for applying their inputs to the entire dataset, in a manner described below. For the remainder of our discussion, we presume that the best two workers are used. Once these best workers are selected, the goal is to utilize their identification of sensitive fields in the representatives they examined to identify sensitive fields in the rest of the dataset.

To do so, we process each new packet not directly examined by the workers by first finding the examined representative that is closest to this packet (i.e., for which the distance (5.1) is the smallest). Pairwise sequence alignment is then performed between the new packet and each representative of the cluster that contains this closest representative. We then adopt the most liberal strategy in marking tokens; that is, we mark a token in the new packet as sensitive if it aligns to a field in any of these representatives that either worker marked as sensitive. We do so because in the domain of packet sanitization, higher recall is typically favored over precision.

5.3 Results and Discussion

In this section, we evaluate the effectiveness of our approach when it is used to identify the sensitive fields contained in packet payloads. For the purpose of our evaluation, we deemed several types of fields as sensitive. These fields were domain names, IP addresses, file names (and directories), user names, passwords, host (server) names, and email addresses. These seven types of fields were used as ground truth for what in the data is "sensitive". We emphasize that these specific fields were chosen only to measure the recall and precision achieved by subjects using our approach. The datasets we used are:

The `UNV-DNS` dataset. This dataset consists of 20,000 network packets recorded at a university campus. The trace contains bidirectional traffic to a DNS server. Of the seven specified sensitive fields, DNS packets contain domain names and IP addresses.

The `KDDCup-FTP` dataset. This dataset was selected from the International Knowledge Discovery and Data Mining Tools Competition.[2] We prepared the dataset by specifically choosing the raw FTP Control packets, which contain 31,020 FTP queries and responses. The specified sensitive fields contained in FTP Control traffic are domain names, IP addresses, file names (directories), user names, passwords, host (server) names, and email addresses.

The `Wireshark-SMB` dataset. This dataset is from the Wireshark^{TM} trace repository. It contains 22,807 SMB (server message block) requests and responses. The specified sensitive fields it contains are domain names, file names (directories), user names, passwords, and host (server) names.

The motivation for selecting these three datasets is that they contain packets with diverse types of sensitive fields and complex message formats. For example, the DNS response packets in the `UNV-DNS` dataset are very diverse and can be quite complex (e.g., with IP addresses appearing in many different places in the response packets). The `KDDCup-FTP` dataset has packets with all the

[2]While this dataset has been criticized as being too unrealistic as a basis for evaluating intrusion detection systems (e.g., [51]), we use it here for a completely different purpose, namely as a source of payload-bearing packets that contain some of the sensitive field types listed above.

sensitive fields specified above, and also has many different types of message formats in FTP reply packets. Similar reasons justify our choice of the `Wireshark-SMB` dataset. For these datasets, we wrote a parser that read the XML packet detail exported by Wireshark to automatically locate all instances of the seven specified fields contained in the payloads. The number and locations of these fields are used only as ground truth.

For the remainder of this paper, we apply standard measures of effectiveness when evaluating our approach, specifically the F-score F_α (see Section 5.2.6) achieved for identifying all sensitive fields either in the entire dataset or simply in the representatives for each cluster. (We will clarify which is used in each case.) Because in the context of packet trace sanitization, recall is often more important than precision — after all, the most common practice when releasing network traces is simply to remove all payload information, yielding a recall of 1.0 but potentially very low precision — we will generally set $\alpha \geq 1$ in our analysis.

In the analysis that follows, we present results from a user study in which professional administrators were recruited to participate, in order to gain a better understanding of the effectiveness of our approach in enabling them to identify sensitive fields. To estimate parameter settings for this study, we first conducted a simulation-based analysis (with no human interaction) to evaluate the effectiveness of propagating marked tokens in the representatives (i.e., tokens identified as sensitive) to the remainder of the dataset.

5.3.1 Exploring the Parameter Space

One advantage of our technique is in generating a limited number of representative packets that capture the characteristics of the packets in the dataset. That said, the manner in which we do so could impact our identification accuracy. Therefore, to choose the most appropriate parameters for our user study (in particular, the number of clusters to use), we performed an analysis in which we simulated a single worker who marked (identified) each instance of a sensitive field independently and with a fixed probability. We reiterate that the sole purpose of the simulation-based analysis was to provide guidance on parameter choice for the field study that followed. With that in mind, we made certain assumptions (about independence) for the simulated user to simplify the task of exploring the parameter space. We then measured the F-score when mapping these random markings of sensitive fields to the full dataset, as described in Section 5.2.6 (though using the inputs of only a single simulated worker, not two in combination). In this evaluation, the simulated worker did not mark non-sensitive fields as sensitive, leading to higher precision than might occur in practice (though the precision on the full dataset was nevertheless always less than 1.0). This was done to focus on the effects of recall or, more specifically, F-score with $\alpha \geq 1$.

We selected 2000 samples from each original dataset using the sampling described in Section 5.2.2. We also controlled the number of representative packets by fixing it irrespective of the number of clusters. Specifically, the number of representative packets was chosen to be 140 in the `UNV-DNS` dataset, 108 in the `KDDCup-FTP` dataset and 120 in the `Wireshark-SMB` dataset; these numbers constituted only 0.70%, 0.34% and 0.54% of the total number of packets in each dataset, respectively. These numbers of representatives resulted from using the technique described in Section 5.2 at the finest clustering (i.e., yielding the most clusters). This number was then fixed as the target number of representatives when fewer clusters were allowed.

Based on these tests, we selected 40 clusters for the tests described in the rest of the paper. For our datasets, this provides a good balance between minimizing the number of clusters that workers

are asked to inspect and providing the opportunity for good accuracy in identifying sensitive data, once worker markings are applied to the full dataset. When selecting the number of clusters in practice, we expect that an expert would be helpful here, as well; that is, the expert can be used to judge the quality of the clustering based on the visual similarity of the packets in each cluster. Thereafter, these clusters can be refined iteratively using the approach presented in Section 5.2.3. Since we observed similar trends for F-scores across different values of α, we settled on a single value of α ($\alpha = 1.2$) when analyzing the performance of real users below.

5.3.2 User Study with Professional Network Administrators

Our techniques for selecting representatives and incorporating user feedback about those representatives to sanitize the full dataset (Section 5.2) are not dependent on any particular method for soliciting that feedback from users. However, we expect that the method of presenting representative packets to users will have a large impact on their abilities to identify and mark sensitive tokens. Two design decisions we made—based on what is known about visual pattern recognition by humans—were to present representative packets as groups based on the clusters to which they belonged (Section 5.2.3) and to present the representatives in their aligned forms (Section 5.2.4).

To determine the impact of these design decisions, and more generally, to evaluate the utility of our overall approach, we conducted an IRB-approved user study with participants recruited from our department's Technical Support Center and the university's Information Technology Service group. All participants were professional administrators with good networking background and familiarity with inspecting packet traces as part of routine network monitoring or diagnostic duties. We targeted professional administrators as they are the natural audience for our tool; after all, they are likely the people who would be tasked with the job of sanitizing network data before its release. This stringent criterion for selecting study participants, however, severely limited the available pool of participants at our university, resulting in our study population of size 15. We note that we obtained consent from these 15 participants only after significant efforts to recruit them. These participants are considered our "workers" in the remaining discussion.

Study Design

Recall that our primary goal was to assess the impact of both clustering and alignment in helping workers uncover potentially sensitive fields in packet payloads. To that end, each worker was tasked with identifying the seven specified fields of interest within the packets displayed via a graphical user interface (see appendix). The study itself comprised four trials in which the payloads of packets were presented to the subjects in different ways. Each trial employed a set $\mathcal{R}$ of representative packets. However, the payloads of these representatives were displayed in different forms in the four trials as follows:

Trial I (Clustering+Alignment). The representative packets $\mathcal{R}$ were partitioned according to the clusters from which they were selected. The representative packets were displayed to the worker, one cluster per page, in their aligned forms produced during their selection (Section 5.2.4).

Trial II (Alignment+NoClustering). The representative packets $\mathcal{R}$ were partitioned randomly into blocks. The total number of blocks was the same as the number of clusters in Trial I, but the representatives were evenly distributed across all blocks. The representatives were displayed to

the worker, one block per page. Since the packets in each block were randomly selected and not aligned with each other, we aligned these packets using the method described in Section 5.2.4 and presented them in that aligned form to the worker.

Trial III (Clustering+NoAlignment). The representative packets $\mathcal{R}$ were partitioned according to the clusters from which they were selected. The representatives were displayed one cluster per page, in their original form (unaligned).

Trial IV (NoClustering+NoAlignment). The representative packets $\mathcal{R}$ were partitioned randomly into blocks. The total number of blocks was the same as the number of clusters in Trial I, but the representative packets were evenly distributed across all the blocks. The representative packets were displayed to the worker, one block per page, with each packet displayed in its original form (unaligned).

For the user study, we chose to use the `UNV-DNS` and `KDDCup-FTP` datasets because they contain diverse types of potentially sensitive information. Two groups of representative packets, one for each dataset, were generated by applying the techniques in Section 5.2 to the two datasets separately. In each trial for a given subject, only one set of representative packets were used, that is, either $\mathcal{R}$(`UNV-DNS`) or $\mathcal{R}$(`KDDCup-FTP`).

Each worker undertook all four trials in individual meetings over a period of several weeks, with at least three days between trials. To avoid any learning effects across trials, we incorporated several additional design elements into our study. First, for the trials taken by each worker, we ensured that the datasets used were evenly split across the trials. Second, to prevent displaying the same representatives on the same page in any two trials for a particular user, we ensured that Trials I and III displayed different representative packets. This constraint was also applied to the two non-clustering trials (i.e., Trials II and IV). Third, the order of the trials was randomly chosen (per subject), and we ensured that no two trials that used the same data (e.g., $\mathcal{R}$(`UNV-DNS`)) were undertaken back-to-back.

Moreover, to limit any factors due to fatigue, the worker was restricted to only one trial per meeting. Meetings were limited to roughly 30 minutes in length, with the exception of the first meeting where the worker was given a brief introduction (with ample time for questions and answers), and time to familiarize herself with the GUI using an artificial dataset. All trials were administered on a dedicated laptop, in a location of the subject's preference. At each meeting, the worker was asked to mark any occurrence of the specified field types, with timeliness as a secondary goal. Care was taken to ensure that the subject was *not* asked to mark content she *thought* could be sensitive, as doing so would be subjective and would inevitably lead to uncertainty about what should, or should not, be marked.[3] That is, her job was to simply mark any tokens in the displayed sequences that she believed to be a domain name, an IP address, a file (or directory) name, a user name, a password, a host name, or an email address.

Results

Figure 5.4 shows box-and-whisker plots of F_α of Trial I for all workers, with $\alpha = 1.2$. Note that these F-scores were computed using the worker's precision and recall on the representatives only, rather than after applying their markings to the full dataset. Each box represents the first, second,

[3]Consider, for example, the username "anonymous" which is not uncommon in FTP; is it sensitive, or not?

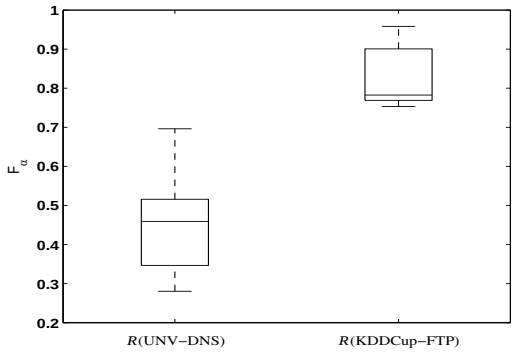


Figure 5.4: F-scores ($\alpha = 1.2$) per worker in Trial I

and third quartiles; whiskers cover the remaining points. Figure 5.4 illustrates that the workers were generally much more successful in identifying sensitive fields in FTP packets, in some cases reaching an F-score exceeding 0.95. The results on $\mathcal{R}$(`UNV-DNS`) were not as encouraging; no F-score greater than 0.70 was achieved by any worker. We believe this reflects the substantial challenge represented by DNS payloads, where the variety of locations in which IP addresses can appear makes identification of such fields a real challenge.

This motivates the need to select only the best workers for identifying sensitive data, and then to employ multiple workers; see Section 5.2.6. For the remainder of our study, we chose the two best workers as determined by their F-scores on a randomly selected 20% of the representatives that each marked. This choice simulates a scenario in which the expert marked 20% of the representatives, and then workers were tasked with marking the remaining 80%. The chosen workers were selected based on their F-scores using the expert-marked data as ground truth. In our tests, we possessed ground truth and so did not need to involve an expert directly.

(a) `UNV-DNS`

Best worker	Recall	Precision	F-score
1	0.504	0.991	0.631
2	0.833	0.674	0.760
Combined	0.900	0.930	0.912

(b) `KDDCup-FTP`

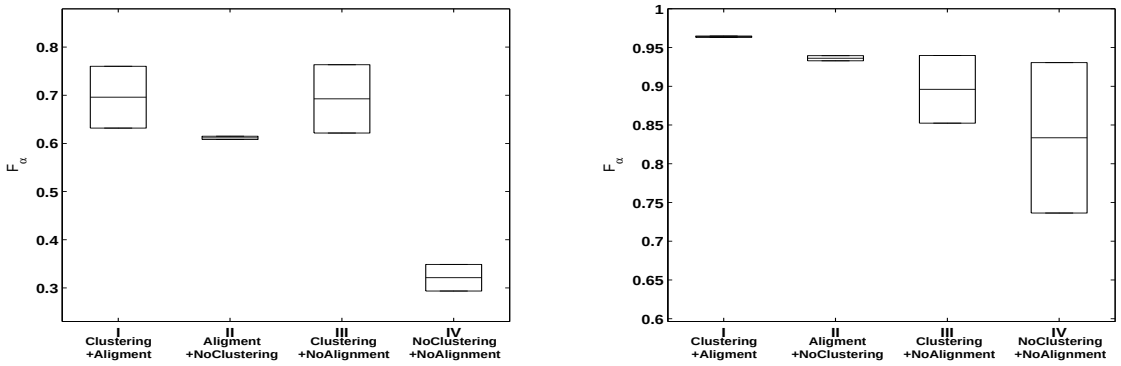
Best worker	Recall	Precision	F-score
1	1.000	0.974	0.989
2	0.958	0.974	0.964
Combined	1.000	0.974	0.989

Table 5.1: Results of applying markings to the full dataset for a single worker and the combined workers

Once the two best workers were selected in this way, their markings were applied to the full dataset as described in Section 5.2.6. Table 5.1 provides the F-scores for the full datasets when applying each of these workers' markings individually and then in combination. As these results show, in the case of the `UNV-DNS` dataset (Table 5.1(a)), the recall of the combined case increases up to 0.9 with a small loss of precision when we incorporate the opinions of the two best workers. For the `KDDCup-FTP` dataset, the measurement of the single worker versus the combined result remains very close (nearly 1.0) because each worker already had high recall and precision on that dataset.

On Understanding Mixed Effects

While the previous results show that substantial improvement in accuracy can be achieved by picking the best workers and combining their input, it is yet to be shown that the clustering and/or alignment aspects of our approach are indeed factors in boosting the workers' performance. To explore the extent to which these two components influence a worker's performance, we first show (in Figure 5.5) box-and-whisker plots of F_α across the four trials for the selected best workers. Notice that Trials I–III generally outperform Trial IV. Notice as well that in Figure 5.5(b), Trial I performs the best, and offers a substantial improvement over Trial IV.



(a) `UNV-DNS` (b) `KDDCup-FTP`

Figure 5.5: F-scores ($\alpha = 1.2$) of the best workers

To gain a deeper understanding of the statistical significance of these trends, we apply a mixed-effect regression model to analyze the four trials of the selected best workers shown in Figure 5.5. A mixed-effect model is an extension of the general linear regression model that allows for correlations within observations [43]. For instance, in our context, this would mean that we consider the performance (say, in terms of efficiency) of a particular worker across different datasets to be correlated, but consider that of different workers to be independent. Conceptually, the mixed effect regression model can be formulated as: $y = \mathsf{fixed\ effects} + \mathsf{random\ effects} + \mathsf{error}$, where the random effects control for variables that are not of particular interest (i.e., the datasets used or different skill levels of our workers), while the fixed effects incorporate the variables that are of interest (i.e., clustering, alignment, and the interaction between the two). The model can be formulated as:

$$y = \beta_c \cdot x_c + \beta_a \cdot x_a + \beta_{ca} \cdot x_c \cdot x_a + \epsilon \tag{5.2}$$

where x_c, x_a are booleans indicating whether clustering or alignment is used, and y is the performance measure under consideration (i.e., F_α or efficiency). The interaction effect of clustering and alignment, i.e., the effect of clustering after alignment is used, or vice versa, is expressed by the product of x_c and x_a ($x_c \cdot x_a$). The random effects derived from workers and datasets are included in term ϵ.

F-Score In the analysis that follows, we first test the null hypotheses $\beta_c = \beta_a = \beta_{ca} = 0$, by fitting all observations of F_α of the best workers using (5.2), the results of which are presented in

Table 5.2(a) with α set to 1.2. We consider p-value < 0.05 as the requirement for rejecting a null hypothesis. As Table 5.2(a) shows, F_α is positively related to the clustering, since the hypothesis $\beta_c = 0$ is rejected and the estimate of coefficient β_c is positive (0.217). Similarly, the alignment significantly increases F_α by 0.197. There is little evidence of an interaction effect $x_c \cdot x_a$ since p-value = 0.123, i.e., the hypothesis $\beta_{ca} = 0$ stands. Even if the hypothesis had been rejected, the estimate of β_{ca} in Table 5.2(a) is smaller (in absolute value) than both β_c and β_a, suggesting that there is an additive effect of these two factors in improving F_α.

(a) F_α, $\alpha = 1.2$

Effect	Estimate	p-value
β_c	0.217	0.011
β_a	0.197	0.017
β_{ca}	−0.160	0.123

(b) Efficiency

Effect	Estimate	p-value
β_c	3.311	0.007
β_a	1.092	0.271
β_{ca}	−2.097	0.147

Table 5.2: Results of mixed-model tests (Section 5.3.2)

(a) `UNV-DNS` (b) `KDDCup-FTP`

Figure 5.6: c^* of best workers and the expert across various θ

Efficiency Yet another important consideration is how clustering and alignment influence efficiency; that is, do they impede or advance a worker's ability to complete the task at hand? Let "efficiency" be defined as $\frac{|\{\text{identified fields}\}|}{t}$, where t represents the total time to completion in each trial. Table 5.2(b) shows the results for a similar hypothesis test for efficiency. The estimate for the clustering term, 3.311, shows that there is a strong, statistically significant, correlation between efficiency and clustering. Although no statistically significant effect of the alignment or the interaction term is found (p-values of 0.271 for β_a and 0.147 for β_{ca}), our tests still indicate that a user's efficiency benefits from both clustering and alignment together, due to the strong positive influence of the clustering.

5.4 Application to Trace Sanitization

A natural question is whether the workers' F-scores were "good enough" to provide a basis for sanitizing the full datasets, and more generally, whether our approach gives enough confidence

to data publishers so that they can release network payloads without the fear of privacy leakage. Answering this question depends ultimately on the data owner's goals for sanitization, which are notoriously difficult to specify or measure for network data, given the difficulties associated with applying existing microdata anonymization definitions (e.g., k-Anonymity [81], ℓ-diversity [48], and (c, t)-isolation [14]) in this domain [22]. To provide a degree of insight, however, we perform a quantitative evaluation of the ability of an adversary to infer the contents of sanitized fields by using a custom, information theoretic measure of privacy.

Recall that our methodology provides a framework for identifying the best workers to mark sensitive fields in packet payloads and applying their markings to the entire dataset. Once this is done, additional steps must be taken to anonymize those values determined to be sensitive, using whatever policies the data publisher desires, e.g., consistently mapping sensitive values to others in a one-way fashion, and perhaps in ways that preserve certain structures (such as a prefix-preserving mapping of IP addresses). For the purpose of evaluation, we employ the same sanitization strategies for all seven types of sensitive fields specified in the user study. Specifically, for each value determined to be sensitive by applying the tokens marked by the best workers to the full dataset, we sanitize this value by deleting it but preserving its type information (e.g., length, text or binary).

5.4.1 Adversarial Model

Our analysis simulates a realistic adversary whose goal is to infer the contents of the sanitized sensitive fields. For the two datasets `UNV-DNS` and `KDDCup-FTP` used in the user study, we simulate a scenario in which data publishers release sanitized packet payloads, and afterwards the adversary tries to recover the real value of each sanitized field. To do so, we presume the attacker can make use of traces collected from the same network shortly thereafter — a very powerful form of attack similar to Ribeiro et al. [71].

To simulate this adversary capability for a particular dataset, we first sort the packets of `UNV-DNS` and `KDDCup-FTP` in the ascending order of time, respectively. Then we assign the first half of the packets into one dataset `SANITRACE`, and assign the second half to another dataset `RAWTRACE`. Intuitively, the splitting of the dataset simulates the scenario that the data publisher sanitized and released one trace of his network collected in a specific period and the adversary acquired another trace of this network collected at a later time. `SANITRACE` is sanitized using the approach described in Section 5.2, while `RAWTRACE` is utilized by an adversary whose main objective is to infer the real values of the sanitized fields in `SANITRACE`.

In what follows, we represent a sanitized packet of
`SANITRACE` as pkt^*. Our simulation methodology first performs sequence alignment (as described in Section 5.2.3) between the sanitized packet pkt^* and each raw packet in `RAWTRACE`. For each sanitized sensitive field fld in pkt^*, we generate a distribution of values aligned with fld from all the packets in `RAWTRACE`. This distribution can be used to measure the adversary's ability to infer the real value of fld. For example, if one value is aligned with fld much more frequently than the others, then this suggests that this value is more likely to be the value of fld that was sanitized. To make this precise, we use an entropy-based measure of this distribution of aligned values.

5.4.2 Entropy-based Measure

Rather than computing the entropy of this distribution directly, however, we adjust this distribution to account for the fact that the packets in RAWTRACE will generally include some that are different in structure from pkt^*; the adversary would presumably give these dissimilar packets less weight in determining the value of fld. Specifically, let $\mathsf{pSet}[v]$ denote the subset of RAWTRACE including each packet that, when aligned with pkt^*, align value v with field fld. Then, we take the probability of fld taking on value v to be

$$\Pr(\mathsf{fld} = v) = \frac{\sum_{\mathsf{pkt} \in \mathsf{pSet}[v]} \mathsf{sim}(\mathsf{pkt}^*, \mathsf{pkt})}{\sum_{v' \neq \mathsf{gap}} \sum_{\mathsf{pkt} \in \mathsf{pSet}[v']} \mathsf{sim}(\mathsf{pkt}^*, \mathsf{pkt})} \tag{5.3}$$

where gap denotes a gap (possibly inserted during sequence alignment) and

$$\mathsf{sim}(\mathsf{pkt}^*, \mathsf{pkt}) = 1 - \mathsf{dist}(\mathsf{pkt}^*, \mathsf{pkt}) = \frac{\mathsf{Score}_{\mathsf{aln}}(\mathsf{pkt}^*, \mathsf{pkt})}{\mathsf{Score}_{\mathsf{max}}(\mathsf{pkt}^*, \mathsf{pkt})}.$$

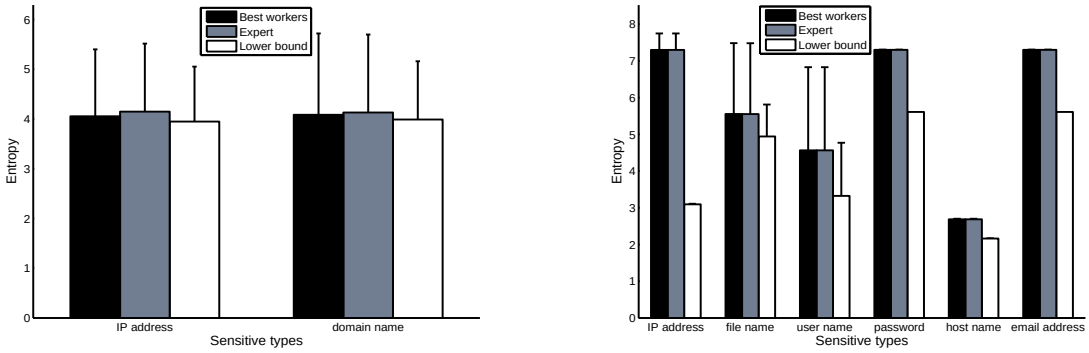
From this probability distribution on the value of fld, the entropy $H(\mathsf{fld})$ of fld is then

$$H(\mathsf{fld}) = -\sum_{v \neq \mathsf{gap}} \Pr(\mathsf{fld} = v) \log \Pr(\mathsf{fld} = v) \tag{5.4}$$

Intuitively, the aforementioned computation measures the ambiguity (from the attacker's point of view) of the value of a field that is masked using our approach. That is, a higher entropy implies more uncertainty in the attacker's inference of the real value of fld. Therefore, the entropy is beneficial in understanding the risk of exposing the real values of sanitized fields to a determined adversary.

We use the entropy as calculated by (5.4) to measure the anonymity of the sanitized fields for the SANITRACE of datasets UNV-DNS and KDDCup-FTP, respectively. Figure 5.7 shows the average entropy of the sensitive fields that are masked by the best workers' marked representatives (the black bars). For comparison, we also show the entropy per (5.4) of the sanitized fields when the representatives were perfectly masked as if by an expert (the gray bars). Although b bits of entropy do not necessarily imply 2^b equally likely possibilities for a field, 2^b serves as an intuitive approximation to understand how many different values the adversary might need to differentiate in order to reveal the real value of the sanitized sensitive field. So, for example, Figure 5.7(a) shows that the average entropy of either the best workers' or the expert's sanitization in UNV-DNS is more than 4 bits for both IP address and domain name, which suggests that from the adversary's perspective there might be 16 different values that appear to be possible for a particular sensitive field. The KDDCup-FTP dataset exhibits similar results in Figure 5.7(b). Also noteworthy is that the workers hid sensitive values nearly as well as the expert did, since the entropy of the sensitive fields masked by the best workers remains very close to that of the expert in both UNV-DNS and KDDCup-FTP.

To place these results in context, we also calculated a rough lower bound on the average entropy that must be overcome by the attacker (the white bars). To calculate the white bars for UNV-DNS, we first separated the packets into different groups (i.e., their respective "formats") according to their record types (e.g., A, NS, MX) based on ground truth. We then computed the entropy of each sensitive field in each group, i.e., of the distribution of values that occur in that sensitive field (at



(a) UNV-DNS (b) KDDCup-FTP

Figure 5.7: The average entropy of the masked fields separated by sensitive types; error bars show one standard deviation

the same packet offset) across packets in the same group. Because these entropies are computed per group, packets need not be aligned and so entropies are not adjusted based on similarities as in (5.4). The white bars for KDDCup-FTP are calculated in a similar way, except that the FTP control packets are grouped by their command types (e.g., "USER", "PASS").

We believe these white bars offer an average lower bound on the entropy faced by the attacker because they are computed with exact knowledge of packet formats, which we do not assume in this paper. In some cases, this lower bound is significantly below the entropy estimated in the absence of that format knowledge (the black and gray bars). The primary cause for large entropy differences is that the alignment between a sanitized packet and each of the raw packets in RAWTRACE (an ingredient in the black and gray bars, but not the white) sometimes led to instances where the values that aligned to a sanitized field included fields of packets in RAWTRACE that are of a slightly different structure. As just one example, in the KDDCup-FTP dataset, sanitized IP addresses in SANITRACE often aligned to *IP address/port pairs* in RAWTRACE, and so the variation in the port values inflated the entropy of the sanitized IP addresses as calculated in (5.4).

These gaps between the white and black/gray bars are particularly interesting because in our calculation of entropy we deliberately attempted to minimize the effects of dissimilar packets by incorporating similarity scores when computing the entropies of values as in (5.4). Nevertheless, the impact of dissimilar packets still exists. An adversary who has detailed knowledge of the packet formats will presumably face uncertainty only to the extent suggested by the white bars on average. That said, as already noted by Cui et al. [27], gaining a full understanding of the structures of packet payloads, especially for protocols that have varied and complex structures, can be quite challenging.

5.5 Recommendations

The fact that our methodology yielded recall of 0.9 and even better precision for the UNV-DNS dataset (see Table 5.1) is, we believe, a very encouraging result, particularly considering the complexity of the DNS protocol. We note that it might be tempting to argue that the results herein could be

improved by permitting workers more time to mark packets; recall that each trial lasted roughly 30 minutes. We believe, however, that permitting more time would yield diminishing returns. Our perception was that packet inspection was a tiresome process for the workers, an observation supported by the fact that none of our participants chose to continue a trial past 30 minutes, though they were given the opportunity to do so. We thus expect that additional innovations will be required to assist workers in identifying sensitive fields more accurately; it is not clear that more time will help.

Our methodology provides a framework only for identifying sensitive fields in packets. Subsequent steps must be taken to sanitize those fields depending on the data owner's goals for privacy protection. Various data sanitization techniques, such as data suppression, psuedonyms and prefix-preserving mappings for IP addresses, can be applied to sanitize the packet trace. In practice, data publishers must choose sanitization policies depending on the levels of privacy and utility they aim to achieve. Advising data publishers on these policies is outside the scope of our work, however.

5.6 Conclusion

In this report we presented a methodology for supporting the daunting task of sanitizing network packet payloads. Our approach is inspired by studies in cognitive science that suggests perceptual grouping of similar patterns can accelerate visual detection. The need for involving humans during the sanitization process is motivated by the complexity of many of today's protocols, and is compounded by the possibility that such protocols may be incompletely documented. However, due to the sheer number of diverse packet formats and the size of a typical network trace, it is unrealistic to expect that an expert will have the time available to accurately sanitize them all. For this reason, our methodology adopts a hierarchical approach in which an expert's input on a small subset of packets is used to select additional workers to examine a larger subset. These selected workers' markings are then used to mark sensitive fields in the (typically much) larger dataset in an automated fashion. At the heart of our methodology is an approach for selecting from the dataset relatively few representative packets for workers to inspect, and presenting these representatives to workers in a way that helps them identify sensitive fields. Our evaluation demonstrated the factors that influence the effectiveness of our methodology, and showed through a user study that our methodology can be effective in supporting sanitization of large network datasets. We also showed, using an entropy-based measure of privacy, that considerable privacy is achieved for packet payloads when applying our framework.

The code and examples are provided in `./Software/DSN11_Amplifying/`.

Chapter 6

Legal and Policy Tool Chest

6.1 Summary

Our Legal and Policy Tool Chest for Cyber Security R&D is comprised of three tools designed to assist cyber security researchers, institutional review boards (IRBs), legal counsel, and others in understanding the legal and policy considerations associated with researchers obtaining and using network communications data1 in cyber security research and development (R&D). The Tool Chest is comprised of three Tools: (*1*) Legal Analysis Tool on Obtaining and Using Network Communications Data (*2*) Privacy Analysis Tool on Using Network Communications Data (*3*) Protection Measures Tool which includes sample Memoranda of Agreement and Clauses and sample Policies. The Tool Chest is designed to assist researchers in determining whether network communications data may be legally obtained and used in R&D projects; Whether privacy laws, legal obligations, and/or organizational policies preclude usage of the data or require that certain aspects of the data be anonymized or eliminated; and what measures can be taken to protect against legal and policy problems in cyber security R&D.

We note that legal frameworks vary globally with respect to the interception, disclosure, and use of network communications data. The Tool Chest focuses on the U.S. legal framework, but mentions important international legal considerations with respect to the privacy of personally identifiable information (PII) and Internet Protocol (IP) addresses. Please see the accompanying 135-page document in `./Legal and Policy Toolchest/`.

Chapter 7

Software and Tools

We provide several tools for achieving the goals outlined in Section 1. The software is provided as part of the included archive. The directory layout is shown in Figure 7.1. It includes all the software, publications, IRB documentation, and several examples of how to use our interactive anonymization tool. The data used in several of our publications is availabe in PREDICT. Pointers to additional third party datasets used in our evaluations is also provided. Please refer to the corresponding README files.

Note that the included software (under `./Software/`) is provided by the authors "as is" and "with all faults." The authors makes no representations or warranties of any kind concerning the quality, safety or suitability of the software, either expressed or implied, including without limitation any implied warranties of merchantability, fitness for a particular purpose, or non-infringement. In no event will the authors or University of North Carolina at Chapel Hill be liable for any indirect, punitive, special, incidental or consequential damages however they may arise. Under no circumstances, including but not limited to negligence, shall the authors be liable for any special or consequential damages that result from the use of, or the inability to use the software. The authors do not warrant that the software will be error-free. The software is provided for research use only.

7.1 Commercialization Plans

Our approach to the commercialization of tools and techniques developed in this project has been two fold; In its initial stages we prototyped technologies and evaluated them using well-established workload and benchmarking suites. Once proven, these technologies were then tested on data collected by our partners at Merit and University of Michigan. The code we developed was shared with independent researchers (e.g., our behavioral clustering technique), but not shared publicly, due to the sensitive nature [1] of the work (e.g., the tools used in [23, 24] can be used to deanonymize network traces). Several of the datasets collected were released in PREDICT — **our main customer**. We currently have no additional commericalization plans for our technologies, but are looking into releasing some parts of the code as open-source offerings.

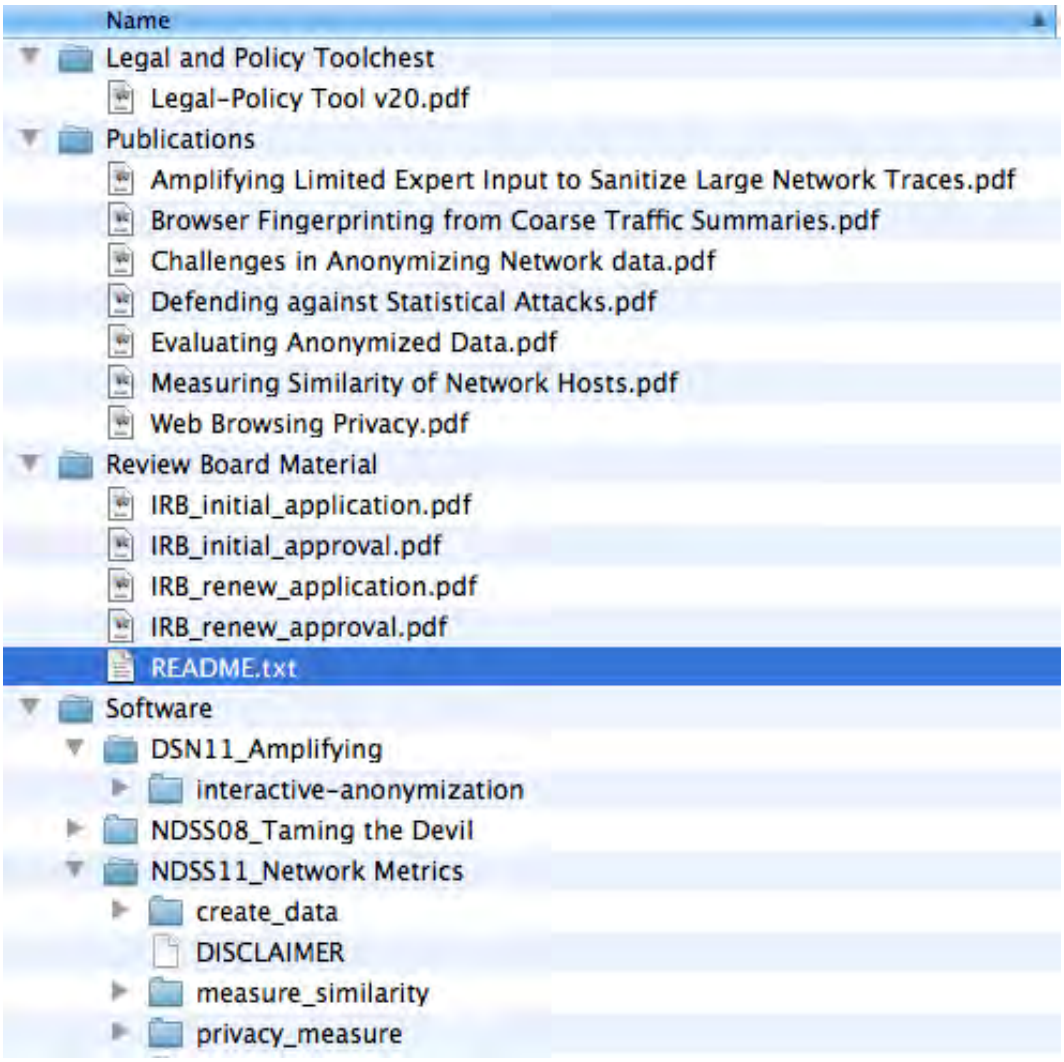


Figure 7.1: Snapshot of directory structure of the included archive

Glossary

APC Uninterruptible Power Supply made by American Power Conversion. 46

CDF Cumulative Distribution Function. 17

CRAWDAD Community Resource for Archiving Wireless Data at Dartmouth. 22

CryptoPAn Cryptography-based Prefix-preserving Anonymization. 17

CSE Computer Science & Engineering. 50

DHCP Dynamic Host Configuration Protocol. 51, 56

DHS Department of Homeland Security. 1

DNS Domain Name System. 7, 56

DTW Dynamic Time Warping. 40

F-score Harmonic mean of precision and recall. 60

FTP File Transfer Protocol. 56

HTTP Hypertext Transfer Protocol. 56

IANA Internet Assigned Numbers Authority. 37

ICMP Internet Control Message Protocol. 44

IP Internet Protocol. 11

IRB Institutional Review Board. 3

JHUISI Johns Hopkins University Information Security Institute. 17

KDDCup-FTP Knowledge Discovery and Data Mining FTP dataset. 61, 62, 64–70

L1 Rectilinear space (also called L1 norm or Manhattan Distance). 9

PII Personally Identifiable Information. 72

PREDICT Protected Repository for the Defense of Infrastructure against Cyber Threats. 1

RAWTRACE unanonymized network trace. 68–70

SANITRACE Sanitized network trace. 68–70

SMB Server Message Block protocol. 56

TCP Transmission Control Protocol. 36

TTL Time-to-live. 36

UDP User Datagram Protocol. 44

UNV-DNS University Doman Name System network trace. 61, 62, 64–70

VM Virtual Machine. 46

Wireshark-SMB Samba traces from the Wireshark Project. 61, 62

XML Extensible Markup Language. 62

XOR Exclusive or. 12

References

[1] M. Allman and V. Paxson. Issues and Etiquette Concerning Use of Shared Measurement Data. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, page To appear, 2007.

[2] D. Arthur and S. Vassilvitskii. k-Means++: The Advantages of Careful Seeding. In *Proceedings of the* 18^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, January 2007.

[3] T. Avraham and M. Lindenbaum. Dynamic visual search using inner-scene similarity: Algorithms and inherent limitations. In *European Conf. on Computer Vision*, 2004.

[4] R. J. Bayardo and R. Agrawal. Data Privacy through Optimal k-Anonymization. In *Proceedings of the 21st International Conference on Data Engineering*, pages 217–228, 2005.

[5] J. Bethencourt, J. Franklin, and M. Vernon. Mapping Internet Sensors With Probe Response Attacks. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 193–208, 2005.

[6] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical Privacy: The SuLQ Framework. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems*, pages 128–138, 2005.

[7] R. Brand. Microdata Protection through Noise Addition. In *Inference Control in Statistical Databases, From Theory to Practice*, pages 97–116, 2002.

[8] T. Brekne and A. Årnes. Circumventing IP-Address Pseudonymization. In *Proceedings of the* 3^{rd} *IASTED International Conference on Communications and Computer Networks*, pages 43–48, October 2005.

[9] T. Brekne, A. Årnes, and A. Øslebø. Anonymization of IP Traffic Monitoring Data – Attacks on Two Prefix-preserving Anonymization Schemes and Some Proposed Remedies. In *Proceedings of the Workshop on Privacy Enhancing Technologies*, pages 179–196, May 2005.

[10] J. Brickell and V. Shmatikov. The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 70–78, 2008.

[11] M. Burkhart, D. Brauckhoff, and M. May. On the Utility of Anonymized Flow Traces for Anomaly Detection. In *Proceedings of the 19th ITC Specialist Seminar on Network Usage and Traffic*, October 2008.

[12] A. J. Burstein. Conducting Cybersecurity Research Legally and Ethically. In *USENIX Workshop on Large-scale Exploits and Emergent Threats*, April 2008.

[13] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in Public Databases. In *Proceedings of the* 2^{nd} *Annual Theory of Cryptography Conference*, pages 363–385, 2005.

[14] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Annual Theory of Cryptography Conference (TCC)*, pages 363–385, 2005.

[15] S. Chawla, C. Dwork, F. McSherry, and K. Talwar. On the Utility of Privacy-Preserving Histograms. In *Proceedings of the* 21^{st} *Conference on Uncertainty in Artificial Intelligence*, 2005.

[16] S. F. Chen and J. T. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech and Language*, 13:359–393, 1999.

[17] Cisco Systems, Inc. NetFlow Services Solutions Guide. `http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.pdf`.

[18] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, Inc., New York, NY, 1977.

[19] S. Coull, M. Collins, C. Wright, F. Monrose, and M. K. Reiter. On Web Browsing Privacy in Anonymized NetFlows. In *Proceedings of the* 16^{th} *USENIX Security Symposium*, pages 339–352, August 2007.

[20] S. Coull, F. Monrose, and M. Bailey. On measuring the similarity of network hosts: Pitfalls, new metrics, and empirical analyses. In *Proceedings of the 18th Annual Network and Distributed Systems Security Symposium*, February 2011.

[21] S. Coull, F. Monrose, M. Reiter, and M. Bailey. The Challenges of Effectively Anonymizing Network Data. In *Proceedings of the DHS Cybersecurity Applications and Technology Conference for Homeland Security (CATCH)*, pages 230–236, March 2009.

[22] S. Coull, F. Monrose, M. K. Reiter, and M. Bailey. The challenges of effectively anonymizing network data. In *Cybersecurity Applications and Technology Conference for Homeland Security*, pages 230 – 236, 2009.

[23] S. Coull, C. Wright, F. Monrose, M. Collins, and M. K. Reiter. Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces. In *Proceedings of the* 14^{th} *Annual Network and Distributed System Security Symposium*, pages 35–47, February 2007.

[24] S. E. Coull, C. V. Wright, A. D. Keromytis, F. Monrose, and M. K. Reiter. Taming the Devil: Techniques for Evaluating Anonymized Network Data. In *Proceedings of the* 15^{th} *Network and Distributed Systems Security Symposium*, pages 125–135, 2008.

[25] T. Cover, J. Thomas, and M. Burns. *Elements of Information Theory, Vol. 1, (revised edition)*. Wiley Series in Telecommunications and Signal Processing, John Wiley & Sons, Inc., 2006.

[26] CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth. `http://crawdad.cs.dartmouth.edu`.

[27] W. Cui, J. Kannan, and H. Wang. Discoverer: Automatic protocol reverse engineering from network traces. In *USENIX Security*, pages 199–212, 2007.

[28] T. Dalenius and S. P. Reiss. Data-swapping: A Technique for Disclosure Limitation. *Journal of Statistical Planning and Inference*, 6:73–85, 1982.

[29] C. D*i*az, B. Seys, J. Claessens, and B. Preneel. Towards Measuring Anonymity. In *Proceedings of Privacy Enhacing Technologies*, pages 54–68, 2002.

[30] J. Duncan and G. Humphreys. Visual search and stimulus similarity. *Psych. Review*, 96:433–458, 1989.

[31] C. Dwork. Differential Privacy. In *Proceedings of the* 33^{rd} *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–12, 2006.

[32] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our Data, Ourselves: Privacy via Distributed Noise Generation. In *Proceedings of Advances in Cryptology–EUROCRYPT*, pages 486–503, 2006.

[33] C. Dwork and K. Nissim. Privacy-Preserving Datamining on Vertically Partitioned Databases. In *Proceedings of Advances in Cryptology – CRYPTO*, pages 528–544, 2004.

[34] J. Fan, J. Xu, M. Ammar, and S. Moon. Prefix-preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. *Computer Networks*, 46(2):263–272, October 2004.

[35] S. Gomatam, A. F. Karr, J. P. Reiter, and A. P. Sanil. Data Dissemination and Disclosure Limitation in a World Without Microdata: A Risk-Utility Framework for Remote Access Analysis Servers. *Statistical Science*, 20(2):163, 2005.

[36] X. Huang, F. Monrose, and M. K. Reiter. Amplifying limited expert input to sanitize large network traces. In *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 494 –505, june 2011.

[37] N. Jones and P. Pevzner. *An Introduction to Bioinformatics Algorithms.* MIT Press, 2004.

[38] A. F. Karr, C. N. Kohnen, A. Oganian, J. P. Reiter, and A. P. Sanil. A Framework for Evaluating the Utility of Data Altered to Protect Confidentiality. *The American Statistician*, 60(3):224–232, 2006.

[39] L. Kaufman and P. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis.* Wiley, Chichester, 1990.

[40] T. Kohno, A. Broido, and K. Claffy. Remote Physical Device Fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 93–108, May 2005.

[41] D. Koukis, S. Antonatos, and K. Anagnostakis. On the Privacy Risks of Publishing Anonymized IP Network Traces. In *Proceedings of Communications and Multimedia Security*, pages 22–32, October 2006.

[42] A. Kounine and M. Bezzi. Assessing Disclosure Risk in Anonymized Datasets. In *Proceedings of FloCon*, 2008.

[43] N. Laird and J. Ware. Random-effects models for longitudinal data. *Biometrics*, 38:963–974, 1982.

[44] K. Lakkaraju and A. Slagell. Evaluating the Utility of Anonymized Network Traces for Intrusion Detection. In *Proceedings of the 4th Annual Conference on Security and Privacy in Communication Networks*, September 2008.

[45] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *Proceedings of the* 23^{rd} *IEEE International Conference on Data Engineering*, pages 106–115, April 2007.

[46] Z. Lin, X. Jiang, D. Xu, and X. Zhang. Automatic protocol format reverse engineering through context-aware monitored execution. In *Network & Distributed System Security Symposium*, Feb. 2008.

[47] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *Proceedings of the* 22^{nd} *IEEE International Conference on Data Engineering*, pages 24–35, April 2006.

[48] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. In *IEEE International Conference on Data Engineering (ICDE)*, 2006.

[49] J. M. Mateo-Sanz, J. Domingo-Ferrer, and F. Sebé. Probabilistic Information Loss Measures in Confidentiality Protection of Continuous Microdata. *Data Mining and Knowledge Discovery*, 11(2):181–193, 2005.

[50] J. P. Mateo-Sanz, A. Martinez-Balleste, and J. Domingo-Ferrer. Fast Generation of Accurate Synthetic Microdata. In *Proceedings of the International Workshop on Privacy in Statistical Databases*, pages 298–306, 2004.

[51] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1998 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and Systems Security*, 3(4), 2000.

[52] A. Meyerson and R. Williams. On the Complexity of Optimal k-Anonymity. In *Proceedings of the 23rd ACM Symposium on Principles of Database Systems*, pages 223–228, 2004.

[53] J. Mirkovic. Privacy-Safe Network Trace Sharing via Secure Queries. In *Proceedings of the 1st ACM Workshop on Network Data Anonymization*, October 2008.

[54] J. Munkres. *Topology*. Prentice-Hall Englewood Cliffs, NJ, 2000.

[55] Nagios IT Infrastructure Monitoring. `http://www.nagios.org/`.

[56] A. Narayanan and V. Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

[57] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[58] Nessus Vulnerability Scanner. `http://www.nessus.org`.

[59] J. Neyman. On two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *J. Royal Stat. Soc. B.*, 97:558–606, 1934.

[60] P. Ohm, D. Sicker, and D. Grunwald. Legal Issues Surrounding Monitoring During Network Research (Invited Paper). In *ACM SIGCOMM/USENIX Internet Measurement Conference*, San Deigo, October 2007.

[61] L. Øverlier, T. Brekne, and A. Årnes. Non-Expanding Transaction Specific Pseudonymization for IP Traffic Monitoring. In *Proceedings of the* 4^{th} *International Conference on Cryptology and Network Security*, pages 261–273, December 2005.

[62] F. Pan, W. Wang, A. Tung, and J. Yang. Finding representative set from massive data. In *IEEE International Conference on Data Mining*, 2005.

[63] R. Pang, M. Allman, V. Paxson, and J. Lee. The Devil and Packet Trace Anonymization. *ACM Computer Communication Review*, 36(1):29–38, January 2006.

[64] R. Pang and V. Paxson. A High-Level Environment for Packet Trace Anonymization and Transformation. In *Proceedings of the ACM Special Interest Group in Communications (SIGCOM) Conference*, pages 339–351, August 2003.

[65] V. Paxson. Strategies for Sound Internet Measurement. In *Proceedings of the* 4^{th} *ACM SIGCOMM Conference on Internet Measurement*, pages 263–271, 2004.

[66] PREDICT: Protected Repository for the Defense of Infrastructure Against Cyber Threats. `http://www.predict.org`.

[67] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[68] T. E. Raghunathan, J. P. Reiter, and D. B. Rubin. Multiple Imputation for Statistical Disclosure Limitation. *Journal of Official Statistics*, 19:1–16, 2003.

[69] J. P. Reiter, A. Oganian, and A. F. Karr. Verification Servers: Enabling Analysts to Assess the Quality of Inferences from Public Use Data. *Computational Statistics and Data Analysis*, forthcoming.

[70] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley. Analyzing Privacy in Enterprise Packet Trace Anonymization. In *Proceedings of the* 15^{th} *Network and Distributed Systems Security Symposium, to appear*, 2008.

[71] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley. Analyzing privacy in enterprise packet trace anonymization. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2008.

[72] C. J. V. Rijsbergen. *Information Retrieval(2nd ed.)*. Butterworth, London, UK, 1979.

[73] D. B. Rubin. Satisfying Confidentiality Constraints Through the Use of Synthetic Multiply-Imputed Microdata. *Journal of Official Statistics*, 9:461–468, 1993.

[74] H. Sakoe and S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[75] P. Samarati and L. Sweeney. Protecting Privacy When Disclosing Information: k-Anonymity and its Enforcement Through Generalization and Suppression. Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory, 1998.

[76] B. Schouten and M. Cigrang. Remote Access Systems for Statistical Analysis of Microdata. *Statistics and Computing*, 13(4):381–389, 2003.

[77] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In *Proceedings of Privacy Enhacing Technologies*, pages 41–53, 2002.

[78] C. Shannon, D. Moore, and K. Keys. The Internet Measurement Data Catalog. *ACM SIGCOMM Computer Communications Review*, 35(5):97–100, Oct. 2005. See `http://imdc.datcat.org/browse`.

[79] Y. Shinoda, K. Ikai, and M. Itoh. Vulnerabilities of Passive Internet Threat Monitors. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 209–224, 2005.

[80] A. Slagell, K. Lakkaraju, and K. Luo. FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs. In *Proceedings of the* 20^{th} *USENIX Large Installation System Administration Conference*, pages 63–77, 2006.

[81] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10:557–570, 2002.

[82] TCPdPriv. `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`.

[83] TCPurify. `http://irg.cs.ohiou.edu/~eblanton/tcpurify/`.

[84] T. M. Truta and B. Vinay. Privacy Protection: p-Sensitive k-Anonymity Property. In *Proceedings of the 2nd International Workshop on Privacy Data Management*, page 94, 2006.

[85] K. Wang, G. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *International Symposium on Recent Advances in Intrusion Detection*, 2005.

[86] M. Woo, J. P. Reiter, A. Oganian, and A. F. Karr. Global Measures of Data Utility in Microdata Masked for Disclosure Limitation. *Journal of Privacy and Confidentiality*, forthcoming.

[87] C. Wright, S. Coull, and F. Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *Proceedings of the* 16^{th} *Network and Distributed Security Symposium*, pages 237–250, 2009.

[88] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proceedings of ACM SIGCOMM*, pages 169–180, August 2005.

[89] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter. Browser fingerprinting from coarse traffic summaries: Techniques and implications. In *Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, July 2009.

[90] L. Zhang, S. Jajodia, and A. Brodsky. Information Disclosure Under Realistic Assumptions: Privacy versus Optimality. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 573–583, 2007.